INL/EXT-21-64448

Light Water Reactor Sustainability Program

Enhancement of Industry Legacy Probabilistic Risk Assessment Methods and Tools

September 2021

U.S. Department of Energy

Office of Nuclear Energy

INL/EXT-21-64448

Enhancement of Industry Legacy Probabilistic Risk Assessment Methods and Tools

Kurt Vedros, Ronald Boring, Stephen Hess, James Knudsen, Svetlana Lawrence, Diego Mandelli, Andrew Miller, Nicolie Aoki, Jooyoung Park, Steven Prescott, and Curtis Smith

September 2021

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy

EXECUTIVE SUMMARY

Probabilistic risk assessments (PRAs) are integral to nuclear power plant (NPP) operations, having tremendously benefitted the safety of the U.S. reactor fleet for decades. Insights obtained from the models have provided perspectives on a variety of applications, both at the plant and for the regulator. While these models are very useful, they are now being asked to represent and analyze aspects of the plant that were never envisioned by the initial PRA practitioners. Furthermore, heightened demands on the PRA models have led to increased computing power requirements. Additionally, as the complexity of the PRA models increased, the difficulty experienced by non-PRA experts in trying to understand these models, grasp the insights they provide, and effectively use that information has become problematic.

The need for research to address key issues regarding PRA tools and methods has never been greater. Although the nuclear power industry has largely been well-served by these tools and methods, the underlying science is dated, remaining mostly unchanged for over two decades. Three areas were identified as most beneficial to address to maintain and improve the usefulness of the current practice legacy PRA tools: improved quantification speed, increased ability to efficiently model multi-hazard models, and improved modeling human action dependency in PRA. This report provides the outcomes from the FY 2021 research into these three areas, identifying potential technical gaps and solutions, and charting the next steps forward to address current PRA challenges.

CONTENTS

FIGURES

TABLES

ACRONYMS

ANN	Artificial neural network
BDD	Binary decision diagram
CAFTA	Computer aided fault tree analysis system
CD	Core damage
CDF	Core damage frequency
CLI	Command Line Interface
CPU	Central Processing Unit
CRM	Configuration risk management
EPRI	Electric Power Research Institute
ER	Equipment reliability
ESREL	European Safety and Reliability
ET	Event Tree
FMEA	failure mode and effects analysis
FPIE	full power internal events
FT	Fault Tree
FY	Fiscal Year
GBWR	Generic Boiling Water Reactor
GPWR	Generic Pressurized Water Reactor
HEP	Human error probability
HFE	Human factors engineering
HPC	High performance computing
HRA	Human reliability analysis
HUNTER	Human Unimodel for Nuclear Technology to Enhance Reliability
JHEP	Joint human error probability
LD	Low dependency
LERF	Large Early Release Frequency
M&D	Monitoring and diagnostic
MBSE	Model-based system engineering
NPP	Nuclear power plant
NRC	Nuclear Regulatory Commission
PC	Personal computers
PRA	Probabilistic risk assessment
PSA	Probabilistic safety analysis

PSF	Performance shaping factors
PWR	Pressurized Water Reactor
R&R	Risk and Reliability
RHR	Residual heat removal
RMTS	Risk-Managed Technical Specification
SAPHIRE	Systems Analysis Programs for Hands-on Integrated Reliability Evaluations
SGTR	Steam Generator Tube Rupture
SPAR	Standardized Plant Analysis Risk
SSC	Structures, systems, and components
SysML	Systems modeling language
THERP	Technique for Human Error Rate Predication
ZD	Zero dependency

ENHANCEMENT OF INDUSTRY LEGACY PROBABILISTIC RISK ASSESSMENT METHODS AND TOOLS

1. INTRODUCTION

Probabilistic risk assessments (PRAs) have advanced the safe operation of the U.S. reactor fleet over many decades. Risk insights from these PRAs have provided information from many different perspectives, from what is most important to maintain at a facility to a better understanding of how to address new information regarding safety issues. The methods and tools that have supported the creation and enhancement of PRA models were established through multiple decades of research, from the 1970s starting with the WASH-1400 Reactor Safety Study [U.S. NRC 1975] through comprehensive plant-specific models in use today.

In August 2020, a report titled "R&D Roadmap to Enhance Industry Legacy Probabilistic Risk Assessment Methods and Tools" was published which outlined key challenges identified by PRA practitioners and detailed research and development (R&D) priorities for advancing the state of PRA software. [Miller, Hess, Smith 2020] From that report, three areas were identified as having near-term strategic benefit to PRA community:

1. Quantification speed of models
2. Multi-hazard modeling development, maintenance, and treatment
3. Human-action dependency analysis.

The report concluded that there are meaningful research activities that can be undertaken to provide near-term benefits to PRA applications and the treatment of models that could push the risk assessment community's state of practice further along. These topics contain well known issues and are quite broad in scope. Further investigation is warranted to determine appropriate tasks and focus areas within each of the broader topics. This will allow resources to be applied to research initiatives that are most likely to have meaningful outcomes and provide tangible improvements to the state of practice.

In addition to determining specific focus areas for each broader area, the research activities must cover software solutions for both U.S. Nuclear Regulatory Commission (NRC) and reactor licensees. The PRA tools used in current practice in the U.S. nuclear power industry are dominated by SAPHIRE for the U.S. NRC and Electrical Power Research Institute's computer aided fault tree analysis system (CAFTA) for reactor licensees. Many ancillary programs are used to support these two PRA platforms; however, the majority of the software's major features include a logic model that reflects the plants design and operation and a quantification engine to solve the model under various postulated conditions for use in various applications. Together, these basic features comprise most of the work conducted by the PRA community on any software platform. For the prioritized research topics indicated above, representative models were identified and utilized in the initial review of each specific focus area. In addition, models specifically used by the regulator and reactor licensee were obtained to test or benchmark performance under real-world conditions.

This report outlines the specific focus areas identified for each broader area, as well as the research, methodology, benchmarking, and conclusions. Because the work described in this report represents initial research in addressing the identified high-value applications, additional follow-on work will be required to fully explore a topic or determine the appropriate scope to realize changes to the applicable methods and implementing software systems. This report serves as a first step in a broader effort to address the legacy PRA software issues and provide suggested upgrades to algorithms, methodologies, and technologies.

2. BENCHMARK MODELS TO BENEFIT TO ALL AREAS

All three areas of interest rely on the capability to perform experimentation using a standard. A search for standard models found that there were few and the ones identified were not gathered into a repository. The team worked with the Idaho State University Nuclear Engineering department to set up a PRA standards repository and identified the initial needs of the repository:

- Generic Pressurized Water Reactor (GPWR) model in SAPHIRE and CAFTA
- Generic Boiling Water Reactor (GBWR) model in SAPHIRE and CAFTA
- Single-top SAPHIRE models of key event trees for pressurized water reactor (PWR) and boiling water reactor (BWR).

Permissions were sought from stakeholders in regulatory and industry and attained over the course of this fiscal year for using existing models to create generic ones. Progress was made populating the repository with the GPWR and single-top SAPHIRE models of event trees in the PWR. The GBWR was postponed until next fiscal year.

2.1 Benchmark Models

The benchmark models created provide structure and content that will allow the quantification of the following metrics:

1. Quantification speed
   - Internal events quantification speed
   - Multiple external events model quantification speed.
2. Multi-hazard modeling
   - Ability to demonstrate handling of high probability hazards.
3. Standard for an external initiating event tree
   - Seismic
   - Event tree based
   - One top based.
4. Standard internal event tree for human event dependency modeling
   - Steam generator tube rupture for PWR
   - A standard event tree for BWR.

2.1.1 PWR Benchmark Model Description

A PWR generic model was developed to be used for training purposes along with research applications. This model is a generic 4-LOOP plant with two trains for support. The model analyzes 10 internal fire scenarios, 11 internal flood scenarios, seven high-wind/tornado scenarios, seven seismic scenarios, and 19 internal event scenarios. This model will be used as part of this research. The results from this model do not reflect any operating nuclear power plant (NPP).

This model is developed into SAPHIRE and utilizes a linked fault tree approach to generating and quantifying minimal cut sets. A linked fault tree approach starts with the event tree logic and links in the fault tree information based on the individual accident sequences. Each accident sequence is mutually exclusive; therefore, all accident sequences can be summed together to get the overall result.

The linked fault tree approach requires assumptions to be made about the failure probability of each top event of the event tree (plant mitigating systems). This assumption is these system failure probabilities are small (i.e., > 0.05) because the success branch is assumed to be close to 1.0; therefore, it

is not accounted in the final result. This assumption is one of the issues identified and discussed in Section 3.

This same model had two of its initiating events, large loss of coolant accident and steam generator tube rupture, converted to a single-top fault tree. This conversion was performed for quantification speed tests and human reliability analysis (HRA)-dependent tests between a linked fault tree approach and a single-top fault tree event approach. The single-top fault tree event approach is the method utilized by a majority of the commercial NPPs. A single-top event is designed to have all of the accident sequences identified via an event tree feed into a single OR-gate. This single gate represents all failure pathways of the plant.

2.1.2 BWR Benchmark Model Description

A BWR generic model will be developed for use in fiscal year (FY) 2022.

2.1.3 Multi-hazard Single-Top Model & Benchmarks

The multi-hazard single-top benchmark model utilized for this project is representative of a typical industry model required for configuration risk management, RICT, 50.69 and other risk-informed applications. The model includes the following hazards for a single unit:

1. Internal events
2. Internal flooding
3. Seismic
4. Fire.

The model comprises the following statistics when solving for core damage (Level 1 PRA):

- Basic Event: 14,343
- Gates: 28,452.

In the benchmark quantification runs, all hazard initiators were included for analysis. The flag files and recovery rules were all standard and consistent across each run. The model was tested with both 32-bit and 64-bit compiled quantification engines to give a sense of performance when moving toward 64-bit code, as may be necessary with complex models or lower desired truncation limits. The benchmark also includes using just the FTREX.exe, as well as the Wrapper.exe to manage multi-threaded quantification. The results are shown in Table 1 with default options selected in FTREX.

Table 1. FTREX results.

Case Name	Truncation	Cut Set	Result	Run Time (min)
CDF_FTREX_2.0_32bit_Wrapper	1.00E-11	92163	2.82E-05	16.2
CDF_FTREX_2.0_64bit_Wrapper	1.00E-11	92163	2.82E-05	14.9
CDF_FTREX_2.0_64bit	1.00E-11	92163	2.82E-05	12.9
CDF_FTREX_2.0_32bit	1.00E-11	92163	2.82E-05	13.4
CDF_FTREX_1.9_32bit_Wrapper	1.00E-11	92163	2.82E-05	13.0
CDF_FTREX_1.9_64bit_Wrapper	1.00E-11	92163	2.82E-05	11.4
CDF_FTREX_1.9_64bit	1.00E-11	92163	2.82E-05	12.5
CDF_FTREX_1.9_32bit	1.00E-11	92163	2.82E-05	12.0

In general, these runs show that the quantification time is similar for all configurations. Utilizing the Wrapper.exe to manage multi-threaded quantification proved to be the fastest for FTREX 1.9 compiled for 64-bit architectures. However, FTREX 2.0 showed an increase in quantification time by 3.5 min with the same options selected. For this benchmark, we will consider all the run times for the cases above to be about the same with a 13.2-min average. None of the standard configuration changes yielded significant reduction in quantification time.

3. QUANTIFICATION SPEED

The single most discussed issue in PRA software is quantification speed. As models have become increasingly complex and required for more applications, so too has the time required to solve a model. Modeling techniques and processes to modularize or pre-solve parts of the tree were common when computing power was relatively low compared to the complexity of a logic model for an NPP. This led to decreases in quantification time but also to a decrease in model resolution as modularized items became obfuscated from the modeler. Over the past several decades, computing power has grown tremendously and the need for these modularization tricks has decreased. Quantification time once again went down but resolution went back up. As computing power continued to grow, more detail was added to the model to support new applications. This was also because quantification speed was not the limiting factor on risk insight production. This situation continued until so much information was added to the model that hardware limits on personal computers (PCs) were reached. This situation has resulted in the current situation that quantification speed for modern comprehensive PRA models (i.e., models that contain internal events, internal flood, fire, seismic, and high-winds information) are prohibitively cumbersome to quantify.

In all PRA software, a logic model is used to represent as-built, as-operated conditions of an NPP. Then, quantification engines (a.k.a. solvers) are used to determine several hundreds of thousands of combinations that lead to an undesired end state then calculate the frequency of that end state. In general, models can have tens of thousands of basic events with many of these basic events having more gates, all of which having an operator. This results in a very computationally expensive process each time a change is made in the model.

The two most common PRA modeling software programs in use in the U.S. are CAFTA and SAPHIRE. In general, the Electric Power Research Institute (EPRI) CAFTA software is used by NPP owner/operators while SAPHIRE is used by the NRC (U.S. regulatory authority). Therefore, in conducting this research, these two software programs are used to explore improvements that can be made in quantification speed.

3.1 Workflow

The basics of workflow of CAFTA and SAPHIRE models are discussed below to help summarize current practice.

3.1.1 CAFTA Workflow

CAFTA is part of the Risk and Reliability suite of software tools developed by EPRI. For NPP PRA models, analysts use the CAFTA Command Line Interface (CLI) for quantifying with the FTREX and QRecover applications. The overall workflow is presented in Figure 3, with specific information on important steps listed below.

3.1.1.1 Requirements for Initial Setup

Installation of the various elements of the EPRI Risk and Reliability Workstation suite of tools is described in detail in documentation available from EPRI. First, install the Phoenix Architect to the default location (e.g., C:\Program Files\EPRI Phoenix\Phoenix Architect 1.0b). Next install FTREX to a

convenient location (e.g., C:\Program Files\FTREX_2_0). Ensure that the FTREX.KEY is copied into this directory. This key is provided by EPRI once a license is obtained.

Next, validate both sets of software. CAFTA can be opened by double clicking the CAFTA.exe file. This immediately launches a dialog asking for a user name, company name, and product serial number. The username can be anything, however, the company name and serial number much match exactly. Validating FTREX has a few more steps. The initial validation must be run with administrator privileges. This is usually accomplished by right-clicking and choosing to run as administrator. The serial number should be provided by EPRI (example serial number shown in quotes) and can be entered using the following command "C:\Program Files\FTREX_2_0\FTREX.exe" /serial=12345678. Alternatively, right-click on FTREX.exe, run as administrator, then enter the serial number when the dialog opens.

3.1.1.2 Logic Model Setup

The installation of Phoenix Architect includes various model files for testing and demonstration purposes. In this discussion, we will use ENO-master.caf file to demonstration quantification with FTREX through the CLI. The demo files can be found in the following directory by default:

C:\Users\%username%\Documents\EPRI Phoenix\Architect\ENO Model

Open the ENO-Master.caf file by double clicking the file. Click the CAFTA icon in the upper left corner, choose Save As, then change the file type to.ftp file. Click save with the same name. Additional files are required for quantification of this model. They are ENO-mex.mutx, SLOCA.flg, and ENO-recovery.recv.

3.1.1.3 Quantification with FTREX

To quantify the model, the following commands in Figure 1 should be used, at a minimum.

```
Call FTREX from Command Line
C:\> cd\Program Files\FTREX_2_0\FTREX.exe /I=%FTPPath%\ENO-Master.ftp /O=%FTPPATH%\Results\ENO-T-006.raw /P=1E-12 /T=T-006 /FLAG=%FTPPATH%\Master.flg /N=%%**
Where
/I = Input File
/O = Output file
/P = Truncation
/T = Top Event
/FLAG = Flag File
/N = Initiator Prefix Pattern
```

Figure 1. Commands for quantification.

The ENO-Master.caf file has a top gate for sequence T-006. Therefore, this quantification will demonstrate quantification of a single sequence, but can be used for any top gate in the model.

The FTREX v2.0 manual states that the output from FTREX can be either a.RAW file (binary) or.CUT file (readable by CAFTA). In our experience, QRecover runs faster when passed a.RAW file to manipulate.

The FTREX manual has numerous options for quantification. The ones shown above are the minimum "standard" options that one could use to get results. However, for large models and models with many initiators, it is often advantageous to group initiators and send those groups to be processed on individual Central Processing Unit (CPU) threads. By default, FTREX will quantify the model without grouping and sending to multiple threads. A separate EXE file called Wrapper manages multi-threaded operations. The default behavior for Wrapper is to split the model by initiators then send each initiator group to a thread. There is no explicit command to manage the number of cores/threads FTREX uses for a given quantification; however, the following commands can be utilized to mimic an explicit command.

The /WRAP_RATIO and /WRAP_MAX advanced commands are placed at the end of the call to send the logic to FTREX, as shown above.

The command /WRAP_RATIO accepts integers not equal to 1. If the user chooses 10 for example, this will group initiators into groups based on the following:

- First initiator will have 1 initiator (first alphabetically)
- The next group will have 10^2 or 100 initiators
- The next group will have 10^3 or 1000 initiators
- This will continue until all initiators are placed in groups (e.g., 10^N).

Each group is then sent to a quantification thread and processing starts concurrently until threads are filled and a new process would need to wait. If this command is used in conjunction with /WRAP_MAX, the user can control the maximum size of the groups. For example, a model with 4000 initiators and a /WRAP_RATIO=1000 and a /WRAP_MAX=1000 would produce five groups with the following number of initiators:

- 1 initiator
- 1000 initiators
- 1000 initiators
- 1000 initiators
- 999 initiators.

These are sorted alphabetically and split into groups. A CPU with four cores and eight threads would utilize 5 threads total to process this FTREX quantification.

3.1.1.4 Post-Processing with QRecover

Some models require post-processing with QRecover. Although FTREX does have a subset of the recovery commands available within its execution, QRecover allows some advanced calls that users have utilized over the years. Therefore, the following command will call QRecover32.exe (or QRecover64.exe) to post-process the output from FTREX (see Figure 2).

```
Call FTREX from Command Line
"C:\> cd\Program Files(x86)\EPRI Phoenix\QRecover32.exe" %FTPPATH%\Results\ENO-T-006.raw %FTPPATH%\REC.recv %FTPPATH%\Results\ENO-T-006_REC.cut
Where
%FTPPATH% is the directory with the files
Note: Qrecover32.exe path is an example using the default location
Optional: /L=%FTPPATH%\Results\ENO-T-006_REC.log
^ This will print the log file
```

Figure 2. Command for the post-process output.

Note that post-processing with QRecover will return a .CUT file. There is no ability to keep the output in a .RAW format.

3.1.1.5 Overall CAFTA Flowchart

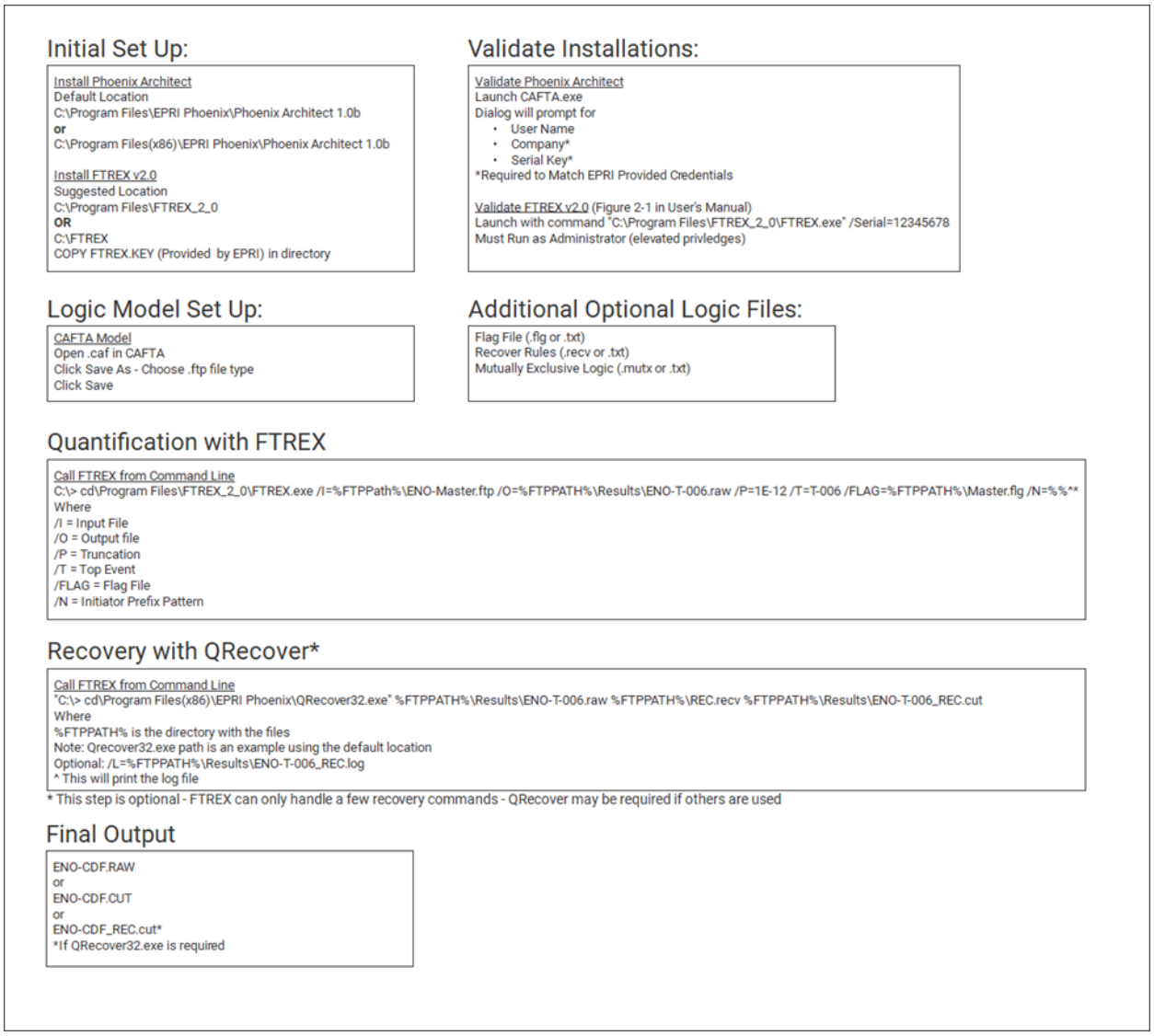

Figure 3. CAFTA flowchart.

3.1.2 SAPHIRE Workflow

Details for how SAPHIRE processes and solves logic models can be found in the associated technical reference NUREGs. [U.S. NRC 2011] For example, the SAPHIRE 8 is found on the NRC website.[a]

[a] https://www.nrc.gov/reading-rm/doc-collections/nuregs/contract/cr7039/v2/index.html

SAPHIRE is a Windows-based program. To start this program, call or click on the SAPHIRE.exe file. Alternative ways to run SAPHIRE are through an application programming interface (API) or through the ability to script functionality through an internal macrolanguage. All logic models in SAPHIRE are stored in a relational database. Once a user asks SAPHIRE to *solve* a logic model (either fault or event tree), the following steps are performed:

- The logic model is loaded from the internal database and restructured as needed. For example, a fault tree may be paged into many smaller fault trees through transfer gates. To solve this model, the logic is connected together to create a single, larger fault tree. Error checking is performed during this step.
- Once the logic model is complete, SAPHIRE expands all N/M gates as needed. N/M gates can be recast as a combination of AND and OR gates.
- Logic loops are situations where the logic loops back to an earlier portion of the logic model, thereby creating a loop. These situations are identified by SAPHIRE so they can be rectified.
- Completed logic gates (either NAND or NOR gates) are converted into non-complemented gates (AND or OR) with the basic events complemented as needed.
- House events (either Ignore, True, or False) are allowed in SAPHIRE to control logic models. These house events are resolved via Boolean logic during this step.
- To speed up processing, SAPHIRE will look for like gates to coalesce these so that like gates are only solved once rather than every time they appear in the logic model. This step looks for modules (shared logic) and independent subtrees (logic appearing only once).
- The level of each gate is determined (top gate is Level 0, the inputs to the top gate are Level 1, etc.) as part of the solution approach. The logic model is reduced and truncated where possible (through size, flag sets, and probability truncation).
- Minimal cut sets are now generated through three processes, gate expansion, Boolean absorption, and cut set truncation. Some parallel processing is used during these processes, up to a maximum of 32 concurrent threads.

After minimal cut sets are produced, they are then quantified using user-selectable methods, such as a minimal cut set upper-bound.

3.2 High-Performance Computing and Multi-Threaded Methodologies

Industry experience with high-performance computing (HPC) and multi-threaded processes used to solve large PRA models has started and stopped, then restarted as personal computer capabilities caught up with the complexity of the models, and then the models became more comprehensive and complex, which necessitated the need for HPC use once again. Industry models mostly use CAFTA. The NRC uses SAPHIRE.

3.2.1 Industry Experience with HPC

HPC usage is not the most common way to solve models for commercial use. Many practitioners elect the convenience of locally stored files, especially during model development, and rarely spend the effort to move, set up, and quantify on HPCs, if they are even available. The exception is some utilities have invested in cluster or cloud computing technologies, but the initial cost of set up generally outweighs the performance gains in quantification time. Additionally, control room operators rarely, if ever, have access to HPCs and developing models for use in configuration risk management (CRM) on an HPC would not yield insights on how long quantification will take when a case is run by operators in the control room that may be time critical.

It is much more common to have a computer with dual processors that can have 10–20 logical cores and threading that allows up to 20–40 simultaneous quantifications to be running. There are several sites in the U.S. that have purchased machines with these types of specifications to be able to determine the risk associated with previously unquantified configurations in the main control room when the need arises. This allows the operators to not have to wait for PRA staff to run and populate the risk metrics for use in decision making.

As mentioned above, FTREX has the capability to manage multiple threads for simultaneous quantification. Theoretically, a computer with enough RAM to support storing calculations and the results should be able to take advantage of multiple threads provided by today's CPUs. A model with many initiations and longer run time per initiators would benefit the most from multiple CPU cores. The code would send the logic model and flag settings to the quantifier to start he quantifications in a batch to have as many threads as available start simultaneously. As runs complete, another thread will start to quantify the model for the next initiator until all initiators are solved. This benefit would be realized for models with lots of initiators and long run times, but real models do not typically act like this.

Significant overhead is associated with creating a new group of initiators for a quantification run. The logic model needs to be reproduced with all other initiators set to false, then quantified, next have the results stored and finally combined with all the other runs. Where this starts to break down is when there are a lot of initiators, but the quantification run time for each individual runs does not take a long time to run. This can happen in fire PRAs quite readily. Models tend to have a lot of scenarios, but the impacts to the tree can be unknown or minimal. In this case, the time required to break each initiator up into a group, set up all the group quantification runs, start the quantification, and combine the results proves costly in terms of processing time outside of quantification time.

The benchmark model demonstrates these inefficiencies. The model has roughly 4300 initiators and there was no real benefit realized using all 40 threads across two CPU cores to run the model compared to single threading the calculation without using the Wrapper.exe. When the software must divide everything into 4300 runs, execute them, and combine the results the time required is large outside of actual quantification. Additionally, many scenarios only took a fraction of a second to compute, so running through 4300 should not take long, yet it still averaged 13.2 min to get results.

3.2.1.1 Quantification Time Reduction Study

Quantification time is a huge topic in the PRA community and understanding multi-threaded behavior, results, and best practices would be a benefit to all practitioners that rely on the model insights for decision making. This study was set up utilizing the same hardware, logic model, flag settings, and recovery rules as the benchmark. However, the difference in settings for FTREX includes utilizing /WRAP_RATIO and /WRAP_MAX to define the number of groups FTREX will make and ultimately send to be quantified using the Wrapper.exe.

When setting /WRAP_RATIO and /WRAP_MAX equal to each other, FTREX will produce the number of groups by dividing the number of initiators by the /WRAP_MAX value. For example, a model with 1000 initiators and a /WRAP_RATIO and /WRAP_MAX set to 250 would create five groups as follows and utilize five threads:

1. 1 initiator
2. 250 initiators
3. 250 initiators
4. 250 initiators
5. 249 initiators.

This specific example would have five quantifications starting at roughly the same time, independent of each other to allow parallel processing. If the settings were changed and the values were 500, it would yield three groups and three processes happening in parallel.

Table 2 indicates the number of cores (and threads) utilized in the quantification run. The number of groups created equals the number of threads indicated. The computer used for this study has 20 logical cores and 40 threads. Therefore, the two case runs that indicate more than 40 threads will create 50 and 100 groups and send 40 groups to the quantification engine, then queue the remaining 10 and 60 groups (respectively) to be sent when quantification of a previous group completes. The resulting times are discussed in the next section.

Table 2. Quantification runs.

Number of Cores Utilized	Number of Threads	Cut Set	Result	Run Time (min)
4	8	92163	2.82E-05	3.3
6	12	92163	2.82E-05	2.8
8	16	92163	2.82E-05	2.8
16	32	92163	2.82E-05	2.4
20	50	92163	2.82E-05	2.5
20	40	92163	2.82E-05	2.5
20	100	92163	2.82E-05	2.6

3.2.1.2 Results of Multi-threading Study

The results of the multi-threading study revealed that unlimited computer hardware does not necessary yield real-world performance benefits for an average PRA model on typical computer hardware. The case with a notably higher quantification time was when using four cores (eight threads) at 3.3 min. Adding two more logical cores (12 threads) yielded a quantification time reduction of about 30 sec. Using two more cores (eight cores), showed no reduction in quantification time. Doubling the core usage to 16 cores, the quantification time dropped by 0.4 min to the lowest case run of 2.4 min. Any additional core or thread assigned beyond this did not yield any reduction in quantification time.

Note that the time steps between four cores, 6-8 cores, and 16 cores are significant. However, within those bands (e.g., 6–8 cores and 16–20 cores) the time differences are most likely due to random variability of the CPU calculations and no real appreciable differences exist.

From these results a significant time reduction was realized from the base quantification noted in the benchmark section. The default FTREX settings, which created a group for every initiator (approximately 4300), were sent each to the quantification engine to utilize a thread per group yielded average run times of 13.2 min, whereas creating less groups with larger amount of initiators yielded an average run time of 2.7 min. This reduction of 10.5 min is a serious improvement in the time required to quantify the model.

These results, although promising, should be expanded to additional models to better understand the relationship between the resources required to create, quantify, and assembly model results by group size and processing time. Quantification times can be highly model-specific and not all models may see the benefits noted in this study.

3.2.1.3 Container Solving

Duke Energy started a process called RapidQuant that utilized the concept of container solving [Corvino 2020]. A container consists of the core code needed to solve part of the problem. Multiple copies of the container are run on servers acting as small computation nodes, which can all work on part of the problem in parallel. RapidQuant split up the solving of the model by initiators and ran each initiator group in a container which performed the FTREX quantification and QRecover process. All cut sets produced by each container were then combined and sent back for results.

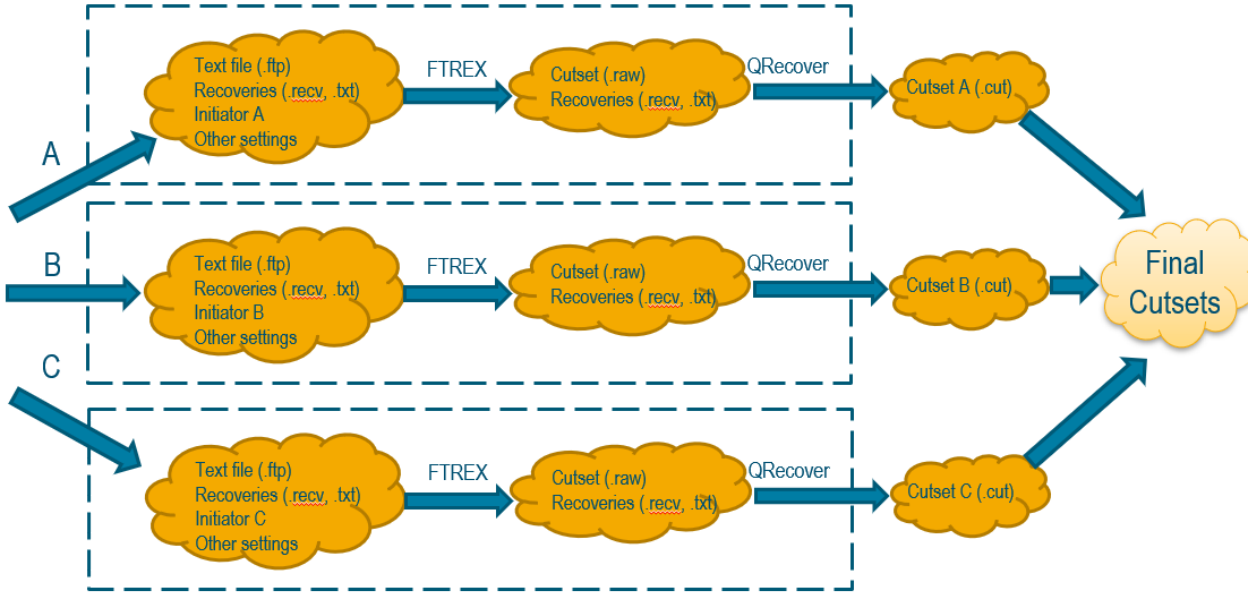

Figure 4. RapidQuant container solving.

This process had a theoretical speedup of 10–12 times. However, the overall benefits were limited by the processing of the slowest initiator. Due to cloud computing security concerns at the time, RapidQuant was not put into use for industry models. Other factors such as further breaking up initiators by mutually exclusive factors optimizing bottle necks could further speed up the process. There is renewed interest in applying this methodology now that more secure networks are available and the legacy PRA project plans to pursue proof of concept studies in fiscal year 2022.

3.2.2 National Laboratory Experience with HPC

3.2.2.1 SAPHIRE Solving Architecture Improvements

INL has researched several optimization options for SAPHIRE involving multi-threading and distributed solving. This led to the current multi-threaded options in SAPHIRE. The limitation of multi-threading is the serial nature of the fault tree and event tree solving and with application of the post-processing. There were marginal quantification speed improvements noted; however, it was determined that the solving architecture could provide more benefits.

Current development for SAPHIRE includes the ability for remote distributed solving through changes in the software and solving engine architecture. The SAPHIRE Solve engine has been extracted from the SAPHIRE software and is prepared to be utilized like an application programming interface (API) that conforms to the representational state transfer (REST) architectural style. This REST API will allow interaction with RESTful web services and in turn will provide more capacity for the solving of fault trees and event trees. This is a major step towards the cloud-based future of the SAPHIRE software. With the SAPHIRE Solve engine now separate from the SAPHIRE software, it can be maintained in a separate code repository and utilized in the popular container solving technology, Docker. Docker paired

with Kubernetes will allow for better solve performance and fast scalability of solving power. Utilizing technologies like Docker and Kubernetes to their full potential only stands to improve performance and user experience using the software. The same docker container can also be used for solving on INL's HPC system for large-scale testing. The HPC cluster analysis can be done to determine the optimal efficiency and bottlenecks. A test version of the docker image was built by the end of July. Full testing using HPC will be done in fiscal year 2022.

3.2.2.2 Use Cases of RAVEN for Parallel PRA Calculations: the SAPHIRE Interface

One of the recent issues regarding PRA maintainability is the required time to perform a full plant PRA calculation once the PRA model is updated (i.e., new data are available, or the actual model is restructured by adding/modifying event tress, fault trees, initiating events or basic events).

While some elements of the full plant PRA calculation can only be performed on a single CPU, few others can be distributed over multiple CPUs by employing computer workstations or HPC clusters.

One of the challenges when dealing with distributed calculations is the management of the data generated on each CPU and the handling of a single job being run on a specific CPU. In the past decade, INL has developed an analysis platform, RAVEN [raven.inl.gov], which allows performance of stochastic analyses (e.g., uncertainty propagation or stochastic optimization) over a given simulation model. One of the major advantages of RAVEN is that these analyses can be performed either on a single or multiple CPUs (e.g., INL and ORNL HPC cluster). RAVEN parallel processing capabilities allow the user to perform data and job management on HPC machines.

Currently, RAVEN can be interfaced with several codes such as RELAP and MELCOR to perform simulation-based PRA. In addition, an interface with the INL PRA code SAPHIRE is also available. The original use case of the RAVEN-SAPHIRE interface was to perform time-dependent reliability analysis where RAVEN is tasked to sample time-dependent basic event failure probabilities/rates and evaluate on the SAPHIRE PRA model as a function of time. This is accomplished by partitioning the time axis into discrete time instances and, for each time instance, basic event failure probabilities/rates are determined and a single SPAHIRE analysis is run.

With the scope of reducing PRA computational time, this interface can be extended to partition the SAPHIRE workflow (Section 3.1.2) into separate steps with the goal to allow RAVEN to manage those steps that can be performed on multiple CPUs. In this scenario, a time-dependent analysis is not actually performed (as originally envisioned for the RAVEN-SAPHIRE interface). However, this interface can be expanded such that it can handle SAPHIRE execution into multiple phases. This would require the construction of multiple SAPHIRE executable files designed to perform specific tasks. Depending on the specification of each task, RAVEN would run the executable associated to this task on a single or multiple CPUs.

3.3 Model Management

3.3.1 Standard Data File Formats

Standard data file formats are explored for the utility of exchange between legacy PRA tools and advanced dynamic tools used to inform the static PRA model.

3.3.1.1 Systems Modeling Language (SysML)

Current plant PRAs are based on event trees (which depict accident progression) and fault trees (which depict failure propagation). One of the PRA-related issues that is being recognized is the complexity reached by these models. Such complexity is reaching not only the actual structure of the event trees and fault trees but also its supporting documents and data sources along with their processing methods. Hence, this complexity is translated into a large number of documents having different kinds of formats (e.g., MS Word docs, pdf files, spreadsheets). A consequence of this situation is that several

version-control and configuration-management issues may arise (e.g., documents shared among different teams, different locations, or different repositories).

A possible solution to this situation is offered by applying model-based system engineering (MBSE) practices to create, modify, and maintain PRA models. MBSE offers a different mindset for the management and control of technical specifications, requirements, interfaces, system design, and analysis. This mindset is based on models (i.e., diagrams). Each diagram represents a specific view (either form or functional) of the considered system. One pivotal MBSE element is the actual modeling language being employed to construct and link the developed set of diagrams. The systems modeling language (SysML) [Friedenthal, 2008] is a generic-purpose graphic modeling language that is widely used for MBSE applications. SysML includes several diagrams (see Figure 5) designed to graphically represent all perspectives of a system (e.g., requirements engineering, system description [i.e., its forms and functions]) and integrate such diagrams with analysis tools (e.g., Modelica).

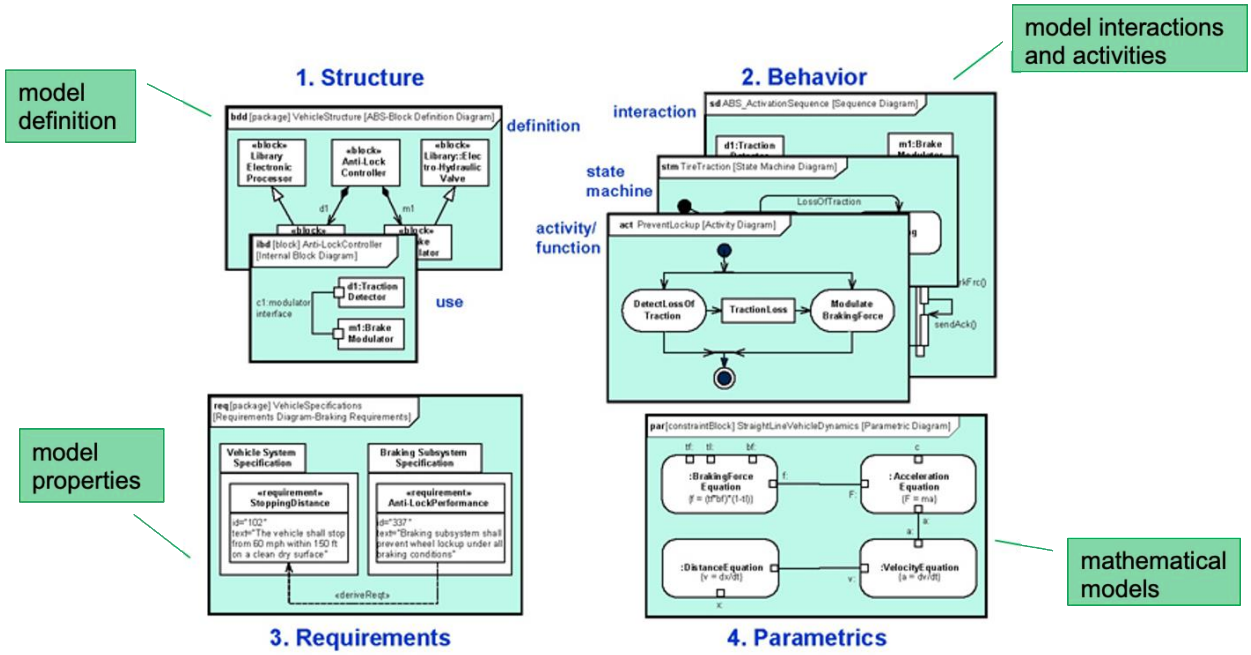

Figure 5. Graphical representation of the diagrams available in SysML (source: http://www.omgsysml.org).

While MBSE practices typically focus on the system design, modeling, and analysis lifecycle, they can be extended to manage plant/system safety, risk, and reliability. The goal is to capture in the plant SysML model the plant form (i.e., system and component decompositions) and functional architecture (i.e., relations and dependencies among components). The use cases for a plant SysML models are as follows:

1. **Automatic creation of reliability models.** As indicated in [Hecht, 2014] and [Rauzy, 2002], from a SysML model it is possible to automatically create system fault trees and failure mode and effects analysis (FMEA) tables. It is here envisioned the analysts would focus on the construction/update of system models in SysML to capture system architecture rather than constructing/updating fault trees or FMEA tables. This would be particularly relevant when the system under consideration is very complex (e.g., digital control systems). The creation of event trees is a quiet different task since it requires system simulation codes to emulate accident progression. An initial approach to automatically create event trees from simulated data is given in [Mandelli 2020] and [Mandelli 2021].

2. **Automatic set up of PRA models provided a user-defined initiating event.** The nature of an initiating event might directly affect a portion of the plant PRA model. As an example, a fire located in a specific room might preclude the function of some components located in the same room. This model dependency can be captured in the plant SysML model in such a way that the definition of the initiating event would automatically set the basic event associated with the affected components to True (i.e., probability set to 1), if the basic event is failure related, or False (i.e., probability set to 0), if the basic event is recovery related. Such component dependency interactions are captured precisely in the activity, state machine, and sequence diagrams (see top right image of Figure 5) that are available in SysML to model interactions among components.

3.3.1.2 Open Source Probabilistic Safety Analysis File System (Open PSA)

Exploration of a common file format for communication between different PRA programs and other software tools culminated in the Open PSA file structure in the early 2010s. Interest has waned in the process recently, but the project team plans an exploration of the utility of building upon the existing Open PSA format for FY 2022.

3.3.2 Use Cases for PRA Version Control Using Git

The past decade has seen the emergence of several platforms designed to perform version-control and source-code management using web-based applications. Two of the most used ones, GitLab and GitHub, are based on the software Git. Both applications are originally designed to support code development by providing project access management and, more important, multiple collaboration features such as issues/bug reporting, code testing management, continuous integration and project information support using the familiar wiki-style [Git, git-scm.com].

However, these same functionalities could be extended to manage plant PRA models over their lifetime. The goal would be to create a unique centralized location of the actual plant PRA model in the current plant information technology services and to use GitLab/GitHub to interface the same model to plant personnel (e.g., PRA analysts, reactor operators, system engineers). It is here assumed that not only plant PRA model is included but also all its relevant documentation (e.g., PRA notebook). The goal of this interface is to manage model modifications both in terms of data (e.g., new reliability data is available) and model architecture (e.g., updated fault or event trees, new initiating events).

Using a Git terminology, the plant PRA model would lie on the main "branch." A direct change of this model would be strictly forbidden. On the other hand, the user that would need to perform such change would follow the following Git version-control path (see Figure 6):

1. **Issue creation.** On the Git interface, the user would create an issue, which indicates the kind of change that is desired to make and the data and/or models that support such change. The created issue is automatically labeled and referenced by Git once it is created.
2. **Creation of a "feature" branch.** The user would then create a branch of the existing PRA model on the Git interface that references the issue created in Step 1. This branch is basically an exact copy of the main branch, and it contains a copy of the plant PRA model. This new branch would exist both in the user local machine but also in the main Git repository.
3. **Commit changes into "feature" branch.** In the "feature" branch, the user is now allowed to perform any modification to the model that he/she deem relevant to the issue created in Step 1 on his/her local machine. The set of changes are then periodically pushed into the main Git repository (commit operation).

4. **Review and integration of proposed changes.** Once the modifications performed in Step 3 are completed, the next step is to "merge" these modifications into the main branch. In a typical GitLab/GitHub setting for code development, the merging operation is allowed only under a specific set of criteria, which include code review performed by a set of peers, verification of the performed modification through a set of regression tests, and creation of new documentation if a new feature is added. Instead, in a PRA setting the desired set of criteria could be the following:
   - Peer review of the proposed changes
   - Execution of predefined data workflows
   - Run complete PRA model
   - Evaluation of the impact of the proposed changes on PRA outcomes (e.g., cumulative distribution function [CDF], Large Early Release Frequency [LERF], risk importance measures).
5. **Merging with main branch.** Once all the criteria have passed, then the proposed changes are merged into the main branch.

The advantage of a GitLab/GitHub framework is that all these operations are archived and can be retrieved to identify the history of changes performed in specific portions of the PRA models.

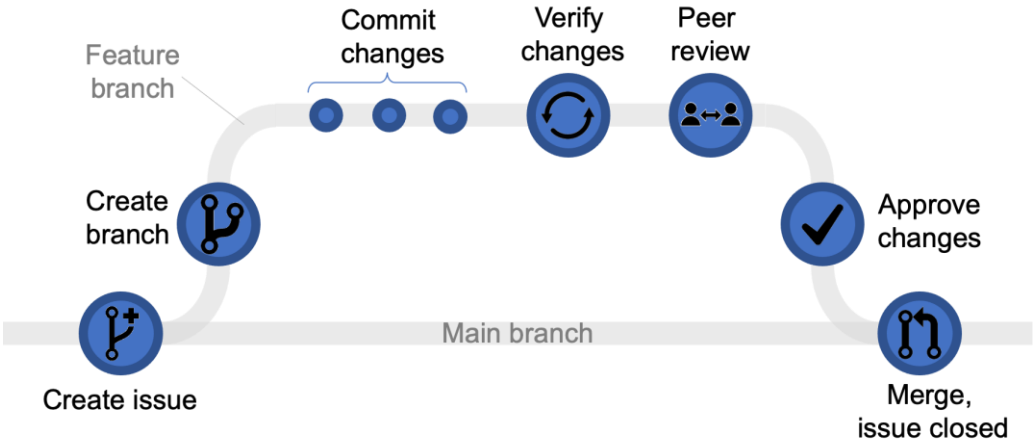

Figure 6. Workflow for the modification of PRA model using GitLab/GitHub framework (adapted from https://about.gitlab.com/).

Figure 7 shows a typical browser-based GitLab interface for a generic project, which provides a wide range of information, such as current number of branches and issues, the repository content, and the people actively editing the repository. From the same interface it is also possible to navigate through active and past branches and issues and identify their content.

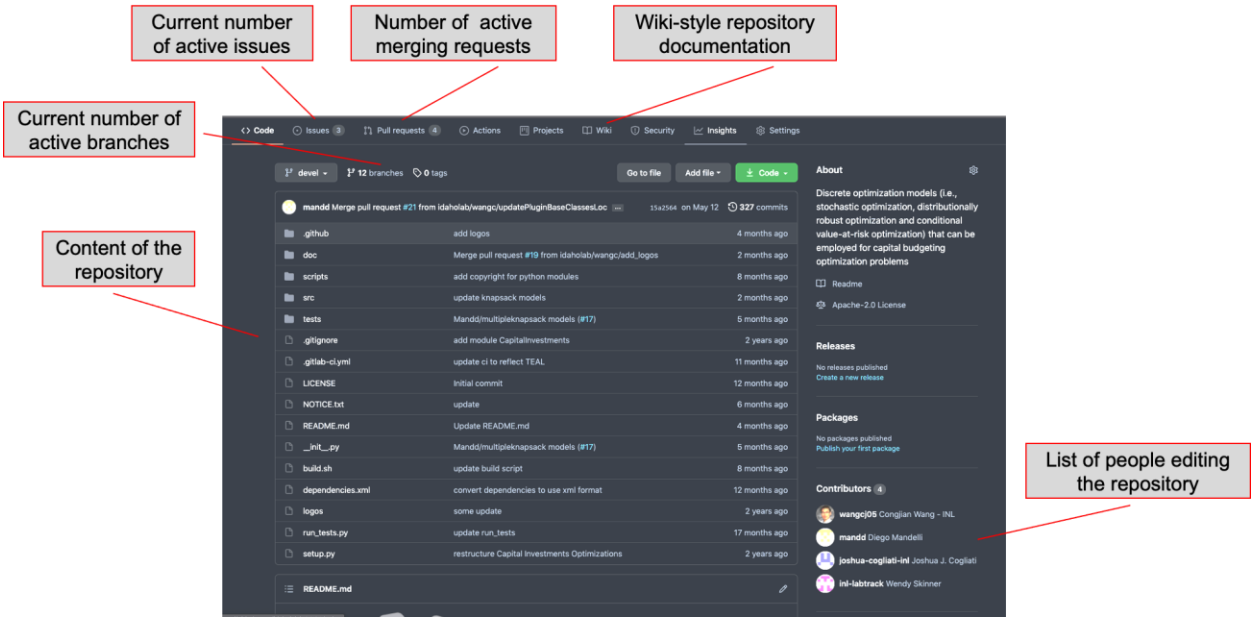

Figure 7. Typical layout of the web-based Git interface.

The use of the GitLab/GitHub framework can be extended to include not only the plant PRA model and its corresponding reliability data, but also all other models that support the plant PRA model itself (e.g., plant thermo-hydraulic model, fuel performance simulation model) and the data workflows designed to analyze and process reliability data (e.g., basic event Bayesian updating). As indicated in Figure 8, not only plant personnel would directly update the plant repository; we give in fact the possibility that actual plant monitoring and diagnostic (M&D) center can automatically update plant data contained in the repository. This would allow a continuous integration of equipment reliability (ER) data with plant models to constantly monitor plant health and its reliability/safety implications.

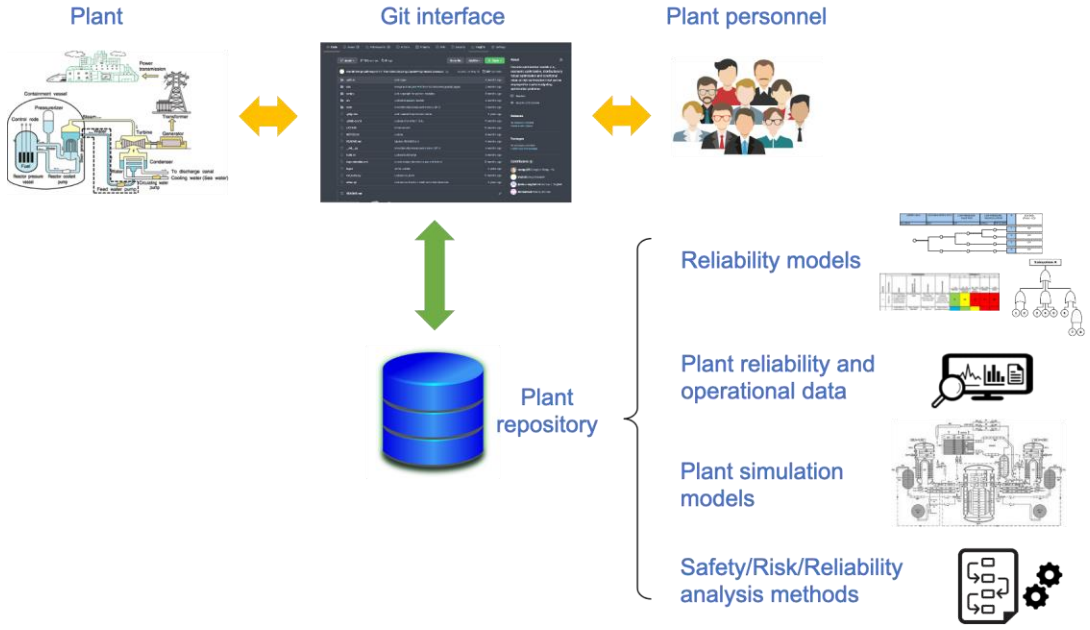

Figure 8. Graphical representation of GitLab/GitHub framework to manage plant models, data, and methods.

Figure 9 shows a tentative tree structure representation of the content of the plant repository, which would include:

- **Models:** All plant-specific models are here included. These models can be either safety related (e.g., input files for safety analysis codes for specific initiating events) or reliability/risk related (e.g., set of event and fault trees for all initiating events).
- **Data:** Here all reliability (e.g., basic event probabilities) and operational (e.g., actual status of components [operational, failed/off-line], and planned maintenance schedule) data is contained.
- **Quantification methods:** This folder would include all the computational engines. These engines can be risk/reliability-related (i.e., PRA code such as CAFTA/SAPHIRE) or safety-related (e.g., REALP5-3D, MELCOR, MAAP). In here, also all the ER data analytics (to generate data required to perform Bayesian updating) and the Bayesian updating routines (to update basic event probability values) would be included.
- **Data workflows:** This folder includes all the data workflows required to verify the acceptance criteria that needs to be checked prior a merging process (see Figure 6). These data workflows would combine models (e.g., plant event trees and fault trees), quantification methods (e.g., PRA code), and data (e.g., basic event failure rates and probabilities) together.

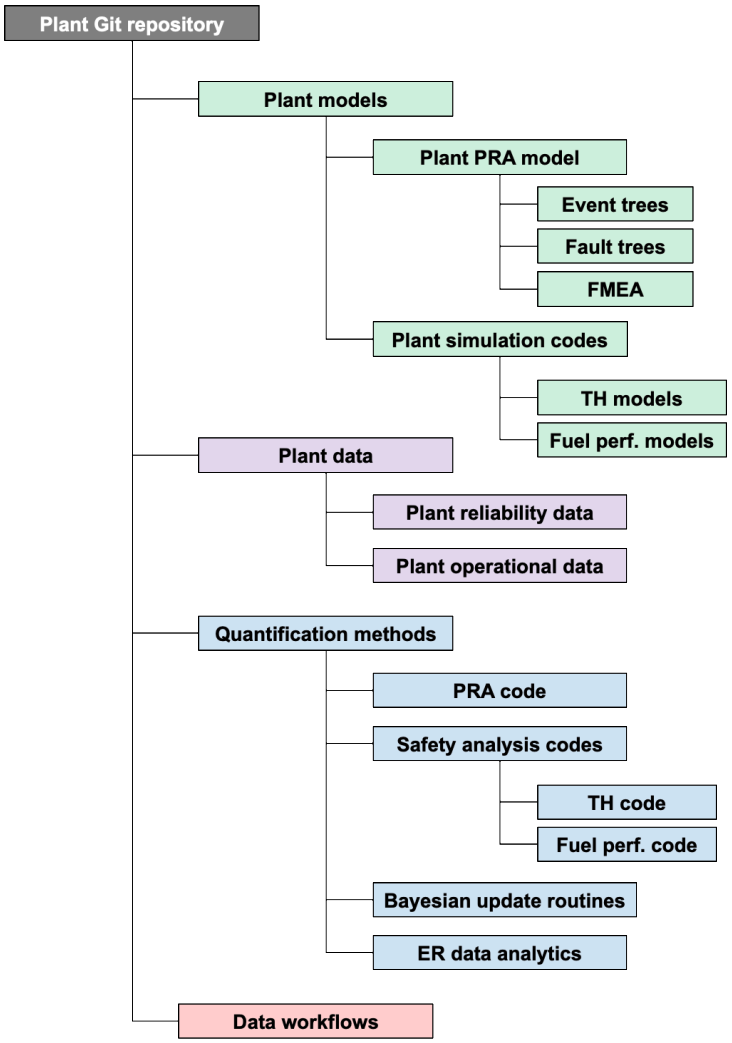

Figure 9. Possible structure of the plant repository.

Examples of automatic data workflows are shown in Table 3. As mentioned above, these workflows would be automatically run every time the content of the repository is modified (see Figure 6). The goal is to create a tractable history of the changes being done in the repository and, most important, evaluate their impact on plant safety and risk.

Table 3. Example of data workflows.

ID	Data workflow	Input Data	Models	Output Data	Dependencies
1	PRA calculation	Basic event probabilities	Event trees and fault trees	CDF, LERF, basic event risk importance measures, set of minimal cut sets	
2	Basic event probability calculation	ER data	ER data analytics and Bayesian updating routines	Basic event probability	Workflow 1 directly follows
3	Event tree validation	-	TH and fuel performance code	Updated event trees	Workflow 1 directly follows
4	Risk monitor update	Plant operational data (logic status of components: operational or off-line/failed)	Event trees and fault trees	CDF, LERF, basic event risk importance measures	
5	ER data analysis	Maintenance, surveillance, or incident reports	ER data analytics	ER data	Workflow 2 directly follows

4. INTEGRATION OF MULTI-HAZARD MODELING

The goals of multi-hazard model improvements are to improve the quantification and presentation of results.

There is a need to describe and demonstrate potential solutions through examples and benchmarking exercises, using both simple and realistic cases. Benchmark models identified for use in quantification speed testing are also used in the multi-hazard modeling improvements testing.

4.1 Multi-Hazard Model Management

The current state of the art in model management is one of brute force and resolving. This served the PRA community well with smaller models, but the addition of multi-hazards and the complexity involved has exposed the limitations of such an approach.

4.1.1 Container Solving

Container solving, as described in Section 3.2.1.3, shows significant promise in quantification speed improvements.

4.1.2 Solving Order and Re-Use

An approach to evaluating multi-hazard models is to solve the Level 1 PRA at a very low truncation. Once this solve is completed, the resultant cut sets are stored. Using the stored cut sets, the different hazards can be quickly evaluated via creation of impact vectors and hazard initiating event frequency.

The process of using these stored baseline cut sets can be:

1. Create an impact vector (i.e., components impacted by the hazard). This would include replacement events to handle conditional probabilities based on the hazard.
2. Assign the hazard-specific frequency for the impact vector.
3. Apply a cut set rule to replace the default components with their updated conditional component probabilities.
4. Re-quantify the updated cut sets based on the impact vector and hazard frequency.
5. Sum up the hazards to get an overall CDF.

The process would eliminate the actual cut set generation for each hazard, which can take a long time to generate the cut sets based on the hazard of interest.

The pros would be the elimination of generating cut sets for every hazard that is to be analyzed within an PRA. This could speed up the quantification process.

The cons would be the initial cut set generation to obtain a set of cut sets that could be used in this process. Another con would be the regeneration of these cut sets every time there is a model change. Lastly, would be the rule process to substitute conditional probabilities for components via a rule process to ensure proper component probabilities.

4.2 Visualization and Communication of Results

Visualization and communication of risk insights is quickly becoming a major part of a PRA analyst's responsibilities. As technology increases and more people become familiar with data visualization, it is expected that complicated tools, like NPP risk models, incorporate a method to effectively convey the most important insights to come out of these models. In this section, two concepts will be addressed as major challenges facing the PRA community on these fronts. The first is distilling lots of information, metrics, and analysis into graphics that can be understood by decision makers (e.g., plant management). The second topic includes challenges associated with incorporating data and results from various hazards that can have drastically different uncertainties, assumptions, and level of model refinements. Both issues play a key role in using and understanding the insights ascertained from risk models and as complexity risk models grow, so too does the challenge of getting the correct information to the decision makers to help guide the choices that need to be made.

4.2.1 Presentation of Results

Modern PRA Risk models have become so complex that understanding the results from the models cut sets has become a highly specialized field within the nuclear power industry. PRA practitioners are now required to understand plant systems and operations, build representative logic models of the plant, quantify the model, and analyze the results to determine the insights required for decision makers. Once the insights are ascertained, presenting them in a concise, effective manner is another issue altogether.

A framework to display the information would go a long way to assisting the PRA engineer in completing their analysis. Reinvention of the visualization graphics each time new results or a new model is created does not prove very efficient. Understanding plant model results "at a glance" can have a powerful effect on comprehension and trust in the model results. Defining visualization techniques that are most appropriate for displaying results and keeping them consistent between model updates and

across a fleet's models can decrease the amount of time to process the results and increase insight intake, as the viewer does not need to recalibrate to get to understanding the results but rather quickly pick up on changes and/or outliers.

A communication dashboard is one example that can achieve this goal. By outlining key metrics to display, fit on a single stylized screen, and maintain a consistent visual style, a decision maker can review one or multiple plants risk insight dashboards to make better decisions in a timely manner. A dashboard can display useful information in an engaging way that feels familiar to the viewer. Standardized dashboards help the viewer to further feel comfortable with complex information because the format remains the same from plant to plant or model to model and all available attention can be focused on the actual insights rather than trying to reassemble results from several plants or historical models to understand what has changed.

Figure 10 is an example dashboard published by NRC. This dashboard displays information regarding precursors at all NPPs in the U.S. At a glance, the viewer can see that precursors per year have been declining through the bar chart in the upper-left corner. Another chart displays the details of the events in the dataset and additional graphics show the number of precursors by plant, risk bin, and when they occurred along with SSC failures. This dashboard demonstrates the power in taking large datasets and creating intuitive graphics to allow for the viewer to process the results.

Another key aspect to visualization and dashboards specifically is user interaction with the data. The dashboard below has several ways to manipulate, filter, and search the data to get specifically tailored graphics that update dynamically when various changes are requested. This level of interaction helps the viewer feel better connected and more engaged with the dashboard and the insights that are being conveyed. A level of customization where the user can manipulate data should be considered in any future research.

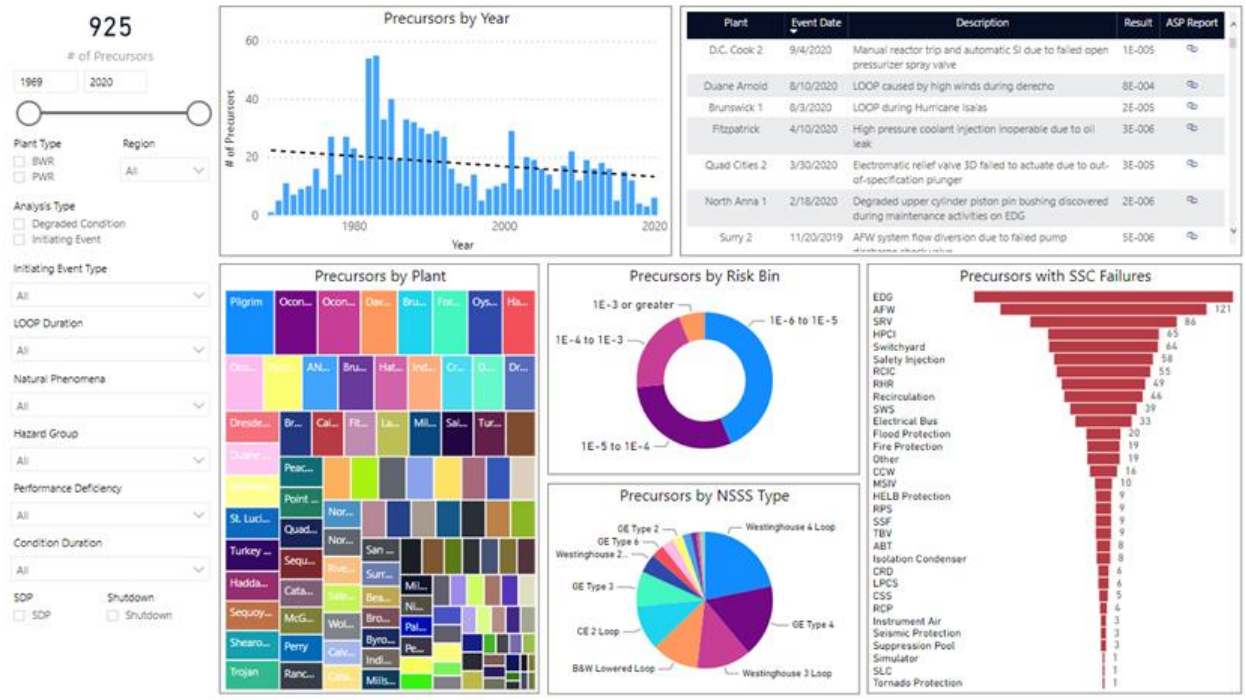

Figure 10. PRA results dashboard example.

4.2.2 Multi-hazard Model Visualization Challenges

Issues can arise when trying to convey information from the PRA as additional models are included in the PRA. It is important to show exactly how things like external events are impacting the risk levels, both in absolute and relative terms, compared to the other portions of the model. Conveying other relative and absolute information, such as uncertainties and importance measures, is important when trying to convey the correct information and to appropriately ensure that the correct message is being transmitted. Proposed improvements in visualization of results moving forward are to create methodologies that outline best practices to ensure the correct types of information are presented in a manner that allows the PRA analysts to communicate proper results from with all available models. Results including assumptions, uncertainties, and relative and absolute values for comparisons should be available at the fingertips of the customer. Visualization in graphical format, such as points with uncertainty bars in Figure 11, is useful to convey a full conceptualization of limits, uncertainties, and sources.

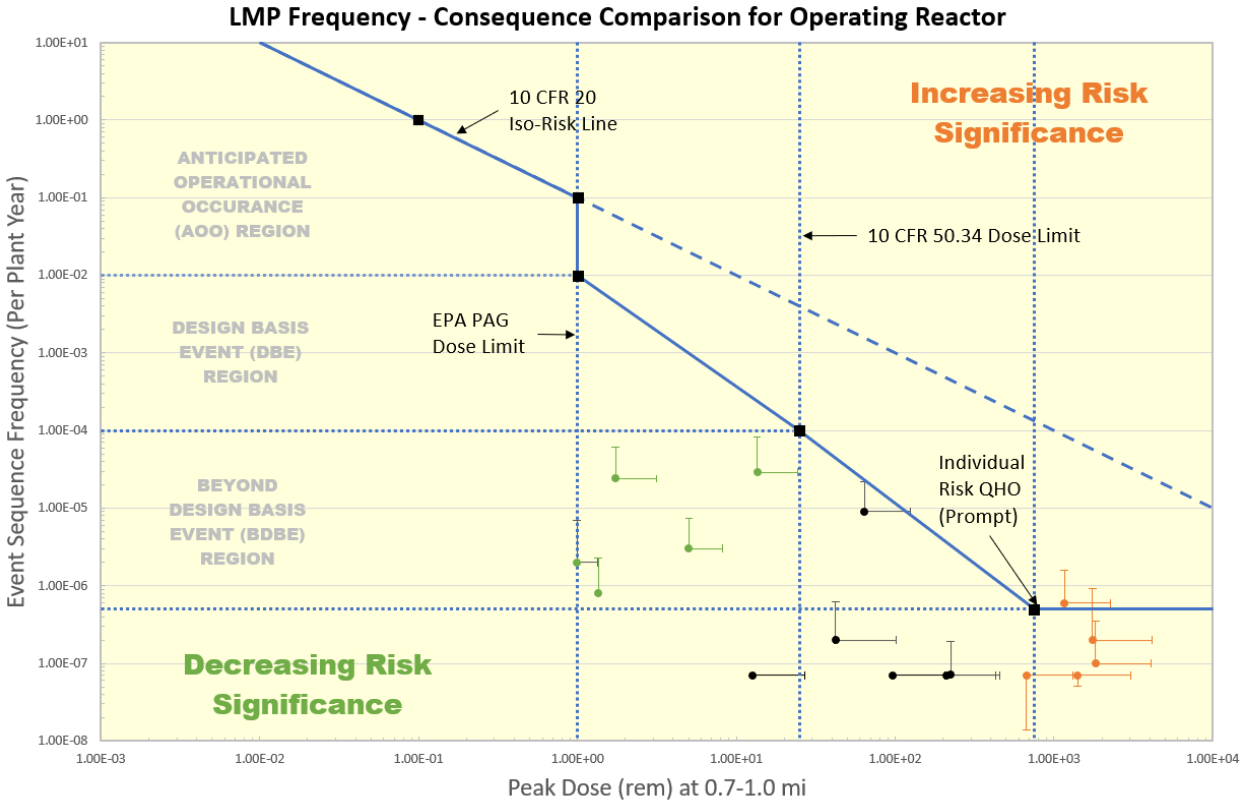

Figure 11. Use of graphics to represent results versus targets (Licensing Modernization Project used on operating reactor).

4.2.3 Visualization Conclusions

The two specific challenges facing visualization of PRA model results and insights warrants further research and pilot application. Testing various visualization techniques, methods, and styles to determine which have the most impact and increase comprehension would benefit practitioners and decision makers. Identifying areas of cross-application between SAPHIRE and CAFTA would also benefit communication between regulator and licensee. One or more of these topics is a suggested path forward for next fiscal year.

4.3 Ability to Efficiently Model Hazard Branches Close to 1.0

External event PRAs contain components that conditioned on the external event could have failure probabilities that are very close to 1.0. These components when solving and quantifying the PRA can cause issues with the final result. Proper handling of these components is very important to not over-estimate the final result. The following will provide a discussion of this issue and provide a potential option to correctly evaluate this issue.

An example of this would be when evaluating seismic PRAs (other hazard PRAs also have this issue [e.g., offsite power recovery]) within an event tree/fault tree method. The event tree logic model will generate accident sequences that need to be solved to obtain minimal cut sets and these cut sets need to be quantified. These accident sequences generate success logic that may need to be evaluated due to the top event failure probability not being a rare event (i.e., greater than 0.05). If the success probability is not properly accounted for at branch splits, an over-estimation of the final result is obtained. For example, if event tree top event T1 in the simple event tree below has a failure probability of 0.3 and the success probability is not properly accounted, then the branch split will evaluate to 1.3 versus 1.0. The success path will assume it is close to 1.0 because of the assumption that all failures of top events are small (>0.05).

The simple event tree shown below along with the discussion will illustrate an approach to handling component failure probabilities close to 1.0.

When solving the accident sequences for any event tree, success terms may be created as part of the Boolean logic and the logic is complemented. Figure 12 shows the event tree that contains systems that either succeed or fail; hence, the arrival of success terms when sequences are solved. For this model, the event tree will be evaluated for Sequence 2, 4, and 5.

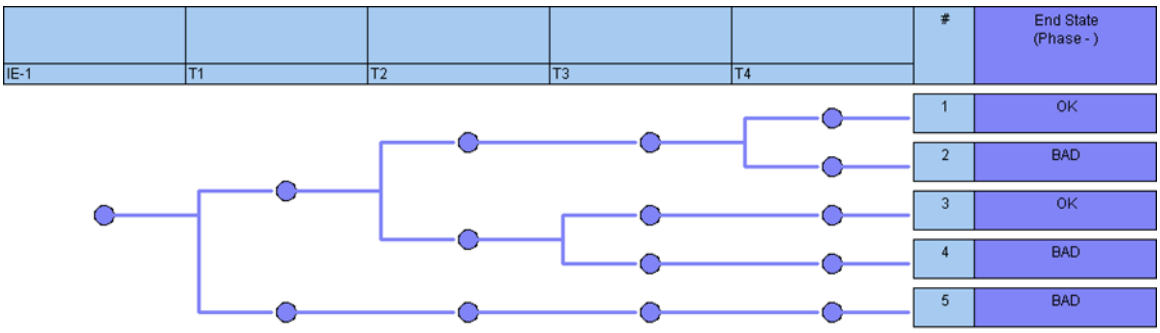

Figure 12. Event tree example with both failed and successful systems.

To solve the accident sequence, PRA software constructs the following logic:

- Sequence 2 = IE-1 • /T1 • /T2 • T4
- Sequence 4 = IE-1 • /T1 • T2 • T3
- Sequence 5 = IE-1 • T1.

The default option to solve these sequences is to use the “delete term” technique to further reduce this list of failed-system cut sets for the specific accident sequence. The delete term technique uses the cut sets from solving the success-system fault tree in the accident sequence logic to eliminate cut sets from the list of failed-system cut sets (the internal AND fault tree). For example, the successful systems fault tree for accident Sequence 4 is T1 (this is the only top that is successful in this sequence).

To perform the delete term step, the PRA software first scans the list of failed-system cut sets and assigns a value of FALSE to any basic event that does not appear in this list. Once this is done, the fault tree representing the successful systems in the accident sequence logic is constructed, pruned by the house events (if any), and solved. Lastly, the final minimal cut sets for the sequence are those that remain after the successful-system cut set terms are deleted (i.e., each failure combination is compared against the success combinations, removing any cut sets that are logically impossible for that particular sequence).

The Boolean logic for the four associated fault trees is shown in below. For this example, all basic event probabilities will set to 0.3 in order to use a high value to demonstrate complications of the quantification process.

- T1 = BE-A + BE-C + BE-B•BE-D
- T2 = BE-C + BE-B•BE-E
- T3 = BE-F + BE-D•BE-G
- T4 = BE-H + BE-E•BE-I.

Table 4 provides some results that can be obtained from different PRA software for the simple example. PRA software will allow for different quantification methods to be used when quantifying cut sets. Depending upon the option chosen along with the cut set generation can provide varied results as shown in Table 4. The two quantification options shown in Table 4 are the standard minimal cut set upper-bound quantification (MinCut) and the binary decision diagram (BDD) [Akers 1978; Rauzy and Dutuit 1997; Sinnamon and Andrews 1997; Andrews and Dunnett 2000].

Table 4. Quantification result for simple example.

Approach		Results		
Cut Set Manipulation	Quantification Approach			
Delete Term	MinCut (using the cut sets) on a sequence, then summing the three sequences for the total	Sequence	Cut Set(s)	Quantification
		2	BE-H + BE-E•BE-I	3.630E-01
		4	BE-B•BE-E•BE-F	2.700E-02
		5	BE-A + BE-C + BE-B•BE-D	5.541E-01
		Summation		9.441E-01
Delete Term	MinCut (using the prime implicants) on a sequence, then summing the three sequences for the total	Sequence	Cut Set(s)	Quantification
		2	/BE-A•/BE-B• /BE-C•BE-H + /BE-A•/BE-C•/BE-D• /BE-E•BE-H + /BE-A•/BE-B•/BE-C• BE-E•BE-I	1.932E-01
		4	/BE-A•/BE-B•/BE-C• /BE-D•BE-E•BE-F	9.261E-03
		5	BE-A + BE-C + BE-B•BE-D	5.541E-01
		Summation		7.566E-01

Approach		Results		
Cut Set Manipulation	Quantification Approach			
Delete Term	BDD (using the cut sets)	Sequence	Cut Set(s)	Quantification
		2	BE-H + BE-E•BE-I	3.630E-01
		4	BE-B•BE-E•BE-F	2.700E-02
		5	BE-A + BE-C + BE-B•BE-D	5.541E-01
		Summation		9.441E-01
None	BDD (using the prime implicants)	Sequence	Cut Set(s)	Quantification
		2	/BE-A•/BE-B•/BE-C• BE-H + /BE-A•/BE-C•/BE-D• /BE-E•BE-H + /BE-A•/BE-B•/BE-C• BE-E•BE-I	1.461E-01
		4	/BE-A•BE-B•/BE-C• /BE-D•BE-E•BE-F	9.261E-03
		5	BE-A + BE-C + BE-B•BE-D	5.541E-01
		Summation		7.095E-01
Exact Answer		7.095E-01		

The simple example above shows that depending upon how the cut sets are generated from the accident sequences and then quantified a varied set of results can be obtained. However, full PRAs are more complicated and the approach above to allow all success terms to be generated and then quantified via BDD is not possible. Therefore, the following approach has been investigated to determine its viability in handling these over-estimations in results.

This approach is identified as “Enhanced Delete Term Assuming Dependent Tops.” This process uses the following steps:

Create the delete term fault trees (i.e., an accident sequence success fault tree representing all the event tree top events that succeed in that accident sequence).

- Solve these accident sequence success fault trees to obtain their failure probabilities.
- Create an accident sequence-specific recovery basic event that takes 1 - Pr(accident sequence fault tree). This basic event will be applied to its specific accident sequence.
- Solve the event tree accident sequences as normal (allow for the default Delete Term option).
- Apply this accident sequence-specific basic event to the sequence failure cut sets.
- Quantify all the accident sequences and then sum up the sequence results.

Applying the approach discussed to the simple example the results are listed in Tables 3 and 4. Table 5 shows the results for the success tops and Table 6 shows the results for the sequence quantification.

Table 5. Success top event fault trees quantification result for simple example.

Fault Tree	Cut Sets	MinCut (1-MinCut)	BDD (1-BDD)
SEQ2-SUC-FT-OR-T1	BE-A + BE-C + BE-B * BE-D + BE-B * BE-E	5.942E-01 (4.057E-01)	5.769E-01 (4.231E-01)
SEQ4-SUC-FT-OR	BE-A + BE-C + BE-B * BE-D	5.541E-01 (4.459E-01)	5.541E-01 (4.459E-01)

Table 6. Sequence quantification results for the simple example and using BDD quantification.

Sequence	Cut Set(s)	Quantification
2	(BE-H) * (4.231E-01) + (BE-E * BE-I) * (4.231E-01)	1.602E-01
4	(BE-B * BE-E * BE-F) * (4.459E-01)	1.204E-02
5	BE-A + BE-C + BE-B * BE-D	5.541E-01
Summation		7.263E-01

5. HUMAN ACTION DEPENDENCY

The goals for improvement in HRA dependency modeling and solving are to represent human dependency efficiently and realistically in the legacy PRA models and tools. Any improvements suggested must be compatible with legacy PRA approaches. There is also a need to describe and demonstrate potential solutions through examples and benchmarking exercises, using both simple and realistic cases. Benchmark models identified for use in quantification speed testing are also used in the HRA dependency improvements testing.

5.1 Current Practice in HRA Dependency Modeling

5.1.1 Description of Common Practices

5.1.1.1 HRA Dependency Modeling Using CAFTA and HRA Calculator

In the U.S., the vast majority of NPP PRA models have been developed using the EPRI Risk and Reliability (R&R) Workstation software suite. Within the R&R Workstation, the CAFTA Event Tree (ET)/Fault Tree (FT) model provides the basic structure for plant PRA models. Due to the specialized nature of evaluation of human actions, a separate tool is used to model and assess human actions; typically, this is performed using the EPRI HRA Calculator software tool. To facilitate these analyses, CAFTA and the HRA calculator have the capability of importing and exporting relevant information between them.

Credit for operator recovery actions typically can be credited only if: (1) the action is performed within the context of an approved procedure and operator training has included the action as part of the crew's training (or that justification for the omission for one or both of these provisions is provided), (2) action "cues" (control room alarms, specific limits of monitored parameters in the plant emergency operating procedures that specify specific response actions, etc.) alert the operator to perform the action, (3) relevant performance shaping factors (PSFs) are assessed and included in the HRA model, and (4) there is sufficient manpower available to perform the action within the required time. Additionally, a recovery action typically can only be credited if it has been demonstrated that the action is feasible for the scenarios to which it is applied and that estimates of probabilities of failure address any dependencies on prior human failures in the scenario.

One of the concerns when conducting an HRA is that the resulting human error probabilities (HEPs) may be too low (i.e., they are beyond the range of credibility). This is a particular concern when multiple human actions must all be performed to achieve a desired outcome (also known as a dependent HEP event). To address this, the HRA calculator has a feature that establishes a lower limit for the total HEP for both individual and dependent basic events (with the default HEP lower limit typically set to 1E-5). For this condition, when the total calculated HEP is less than the lower limit, the software changes the probability from the calculated HEP to the lower limit selected. HEPs that are modified in this manner are tagged so they can be identified by the analyst as "Below HEP limit" in the HRA calculator summary view under Additional Info, as well as in reports generated by the software.

There are several approaches for use of the HRA calculator to evaluate dependent events and integrate them with the PRA model. Most of the work to accomplish these objectives is performed in the HRA calculator. The following represent key steps that typically are performed:

1. Identify relevant combinations and create recovery file.

   The first step is to identify combinations of human failures that have a significant impact on PRA model results. The typical approach is to first solve the PRA model with all HEP values set to 1.0 (or some other suitably high probability) and generate cut sets, which will now contain representative HEP combinations. Setting the HEPs to an artificially high value ensures cut sets containing the human factors engineering (HFE) combinations to be processed for dependency do not fall below the truncation value prior to being analyzed and applying an appropriate value to the combination of events. As mentioned above, combinations of events can often mathematically calculate to a probability below limits that are considered to be credible and could erroneously fall below the truncation value and then be excluded from the dependency analysis altogether.

   The HRA Calculator Helper is then used to evaluate the combinations of HEPs that appear together in all the cut sets generated after quantification. The HRA Calculator also creates a recovery rule file to be used in post-processing the cut sets. This is created based on the HEP events and their combinations that are included in the HRA Dependent Events Report and are written to a QRecover Rule File. The recovery file typically performs the following actions after initial quantification of the model:

   - Sets all independent HEPs to 1
   - Adds dependent HEPs via one of the following methods:
     – Adds an event to the cut set that represents the dependent HEPs from the dependency analysis and setting the remaining independent events to 0 or false.
     – Replaces all the independent events with a single dependent HEP event (in this case, independent events are no longer visible in the cut set).
   - Sets non-dependent events back to their original value.

     After the recovery file is created, the new combination events are added to the CAFTA database file.

2. Perform model quantification.

   FT models can be evaluated via multiple approaches in CAFTA. One can either calculate probabilities for every gate in the tree or generate cut sets for the top event using the cut set generator. Additionally, CAFTA can be configured to use third party evaluation tools if desired.

   This quantification must be performed with the HEPs set to artificially high values to ensure all the relevant events are included in the cut sets without falling below the truncation value, as described above.

3. Implement recovery rule file.

Once model quantification is completed, the recovery rule file is applied to the cut set file. The results of the final model quantification and recovery process are then reviewed, and modifications are made by the analyst. It is noted that the CAFTA documentation recommends that analysts should not change the master recovery rule file; rather, they should add a link to the new recovery rule file inside the master recovery rule file. However, this is rarely done in practice.

It should be noted that all model changes require repeating of this entire process. Even small adjustments to the logic model can yield impactful changes to the HRA dependency analysis. There really is no current way to bypass repeating this time consuming, multiple-step process described above when working to update or change aspects of the model.

Results of HRA models using the HRA calculator can be exported directly to CAFTA. Alternatively, the HRA calculator can export to an Excel.csv format file. For the direct export function, the user has the option of exporting all HRA events or specifying a user selected subset. For these exports, the selected HRA events included in the.HRA file are exported to the selected CAFTA RR file. Additionally, the user can select to export mean values, 95-th percentile values, 5-th percentile values, or 1.0 values for the HEPs to the designated RR file. (If the basic Event ID does not exist in the CAFTA file, a new basic event is added. If the basic Event ID does exist in the CAFTA file, all fields are updated.)

5.1.1.2 HRA Dependency Modeling Using SAPHIRE

Dependency in SAPHIRE is applied chronologically. The first operator action along the sequence is left as is and then a dependency analysis is performed on the second, and a third operator actions within the sequence, if necessary.

For example, consider RCS-SGTR (operator fails to initiate SGTR procedures) * RHR-ERROR (operator fails to initiate RHR cooling).

Since the first operator action is RCS-SGTR, the independent probability is left as is and then, based on the SPAR method, the RHR-ERROR operator action is adjusted based on the failure of RCS-SGTR. The combinations are identified from the cut sets obtained from the model quantification where individual HEPs are set to a higher value (e.g., 0.1–1.0). Once combinations are identified, the dependency analysis is performed for each combination for each pair of the operator actions. The HEP for the second action in the pair is adjusted for dependency based on the SPAR-H dependency rules. The dependency is applied with post-processing rules.

Final model quantification is performed with the nominal HEP values. The concern with this approach is when the model is solved with all HEPs back to their nominal values, some of the combinations could be truncated and the dependency will not be applied, thereby missing some operator action combinations in the final cut sets.

5.1.2 Challenges in Current Practice of HRA Dependency

In this section, we review some of the challenges with performing human reliability dependency analysis. These challenges can be delineated into implementational challenges (primarily involving software applications) and general challenges for how to treat HRA dependency. We begin with a discussion of challenges in the current practices of CAFTA and SAPHIRE HRA dependency modeling. We then discuss general challenges, which set the state for future research activities.

5.1.2.1 CAFTA Challenges

Issues identified with the current state of practice of HRA modeling in CAFTA are:

- Sequence of operator actions is not available. In many cases, the sequence (chronology) of the few actions in a given cut set is not easy to interpret. A specific “walk-thru” is required with operators to properly place HFEs in order in a given combination.

- Usually, there are thousands of combinations identified in cut sets with initial HEP=1.0. It takes time to post-process recovery rules after dependency analysis is done. In some cases, there are 10,000+ combinations.
- If properly done, HRA dependency is an iterative process where a balance is sought between too many and too few combinations (rule of thumb 1,000 to 3,000). This iterative process takes time.
- The dependency analysis is very subjective (analyst-dependent) and many "overwrites" are usually applied to "downgrade" auto-determined dependency level.
- The need to have initial HEP=1.0 (or high-value) creates confusion, model maintenance issues, and human errors. The baseline PRA model does not have HEP=1.0, it has nominal (independent) HEPs. The high-value HEP model is created for quantification only to identify as many combinations as possible to replace them with the dependent HEP value (or the minimum floor value).
- Dependency needs to be re-analyzed for hazard-specific scenarios where some independent HEPs are adjusted to account for hazard-specific conditions (higher stress, smoke for fire scenarios, etc.). This additional dependency evaluation generates another set of hazard-specific combinations (i.e., even longer recovery rule file).
- The interaction between CAFTA and EPRI HRA Calculator is semi-manual (i.e., import values from CAFTA, export values to CAFTA). There are two databases (CAFTA and Calculator) that create a potential of model record issues and a need to re-run dependency analysis every time any HFE changes probability (need to re-create the recovery rule file with the new dependent HEP values even if nothing in the dependency analysis has changed).

5.1.2.2 SAPHIRE Challenges

Issues identified with the current state of practice of HRA modeling in SAPHIRE are:

- Most of the processes in HRA dependency analysis may be treated outside of each PRA software. But they support how to reflect the result. In SAPHIRE, the result is applied using post-processing rules and the dependency quantification option in "Edit Basic Event" window.
- SAPHIRE can extract dependency candidates (i.e., HFE combinations) from cut sets using "Slice" option, but there is no worksheet to evaluate dependency levels within the software. The worksheet is especially required when analyzing dependencies among more than three HFEs.
- The software does not provide a function or a worksheet for arranging HFE candidates in a scenario order or actual scenario timeline. Because a cut set sequence generated from a PRA tool does not always indicate a scenario order or actual scenario timeline, it needs to be adjusted before analyzing dependencies between HFEs.
- The SPAR-H dependency factors in SAPHIRE look like a little bit confused (e.g., "Crew"=> "Same" or "Different" in SPAR-H but "Different Crew"=> "Yes" or "No" in SAPHIRE). The dependency logic tree is not also shown in the sheet. Some practitioners use the logic tree to review the result of dependency analysis.
- There is no clear function to analyze dependencies among more than three HFEs and apply a couple of additional dependency evaluation rules suggested in SPAR-H.
- There is no publishing function for the results of dependency analysis. The HRA dependency analysis highly depends on analysts' judgement. The entire process must be more traceable.

5.1.3 General Challenges

A number of challenges are more foundational to HRA and supersede the issues with specific tools used for dependency. These challenges include:

Existential challenges related to dependency. Almost all treatment of HRA dependency stems from the original HRA method, the Technique for Human Error Rate Predication (THERP) [Swain and Guttmann 1983]. This approach postulated direct dependency, in which subsequent error is more likely following an initial error. One formulation of this is "error begets error," although the treatment of error in THERP does not suppose that one error is necessarily caused by another error. Mortenson and Boring [2021] have taken up the argument of whether dependency actually exists. While Swain and Guttmann created a compelling case for the need to consider dependency and offer a mathematical treatment to reflect increased error propensities after initial errors, the authors point out that this concept does not exist elsewhere. Indeed, the idea of dependency remains fully rooted in risk literature but does not have corresponding evidence in psychological research. The authors suggest before tackling the implementational details of HRA dependency, it may first be necessary to establish it empirically through observational studies.

Indirect dependency. [Park and Boring 2021] review the literature surrounding indirect dependency. This form of dependency suggests that it is not the occurrence of a human error that primes the subsequent occurrence of errors. Rather, it is a common context that triggers more than a single human error over time. This context can be captured in PSFs, which are already treated by many HRA methods. However, these methods do not include a method to account for increased error rates common across PSFs. Mostly human failure events are treated as distinct entities for analysis, and the carryover effects of PSFs across human failure events are not addressed. This phenomenon was articulated as PSF "lag and linger" [Boring 2015], meaning some PSFs do not invoke instantly but rather build up in effect over time (i.e., PSF "lag") and some PSFs do not cease instantly but rather continue to have a gradual degradation over time (i.e., PSF "linger"). These concepts have been demonstrated in limited use but require further validation. Indirect dependency must be treated differently than direct dependency. It is not a correction that is applied based on judgement of dependency but rather intrinsically linked to the error mechanism and how it propagates over time. Indirect dependency is part of the very calculation of the basic human error probability and not a separate treatment after the fact.

Formative dependency. Formative refers to something done early in a process at a stage where it helps *form* the outcome. The concept of formative dependency [Boring, Park, and Mortenson 2021] refers to applying dependency corrections earlier in the calculation process. Because dependency strongly anchors the value of the human error probability to a few set values, this process proposes to forego regular calculation of the human error probability and instead only use the dependency anchor value from THERP as the human error probability. This approach is presumed to provide computational advantages by skipping calculation several steps that may ultimately wash out when applying HRA dependency.

Dynamic dependency. In dynamic treatments of HRA, one approach to calculating HEPs may see them calculated at a finer granularity than the typical human failure events used in HRA. This subtask level of analysis poses challenges for the application of HRA dependency corrections because it may risk artificially inflating the resulting HEPs when they are aggregated [Boring 2015]. There are other treatments of dynamic HRA that may calculate success and failure based on the outcomes of incremental simulation runs. The outputs of these runs need to be informed by dependency to run successfully. In other words, if dependency is present, it is assumed there will be an increase in the failure rate. Whatever the treatment of HRA dependency, as dynamic HRA methods are made more mainstream and come to support more industry applications, there remains a strong need to determine the mathematical treatment of HRA dependency for simulated courses of human actions.

Dependency validation. Most of the above challenges are a subset of an overriding challenge—that there has not been much empirical research performed to validate HRA dependency. Without formal studies to review whether error begets error, it is impossible to determine the degree to which HEPs are influenced by preceding errors. Empirical research on HRA dependency—of errors in sequence—

can establish the extent to which dependency should be considered in HRA models and confirm the mathematical correction factors used to drive up HEPs in the face of dependency.

5.2 Recommended Improvements to Current HRA Dependency Modeling Approaches

Both CAFTA and SAPHIRE approaches model and solve HRA dependencies at a low level. One potential improvement is to move the solving to a higher level within the model.

5.2.1 Perform Dependency Analysis Before Quantification

Dependency analysis is usually done after the model is quantified. The combinations of operator actions are identified from cut sets (i.e., the quantification result), where a combination represents a set of operator actions found in an individual cut set. A dependency analysis is performed for the actions within a given combination. The HEP of a subsequent action is increased if it is determined that dependency exists.

The proposed approach is to evaluate dependency between the actions before the model is quantified. The operator actions in this case are identified from accident scenario sequences using ETs as the timeline to sequentially order the actions.

5.2.1.1 Approach Benefits

This approach eliminates the need to manually order operator actions sequentially (actions returned form the cut sets are in random order, not sequential). This is a vital task in the dependency analysis since the dependency is addressed between two actions in time (i.e., second action is assessed if it is dependent on the first action). Change of the action order would cause inappropriate dependency analysis and dependency level. The sequence of the actions is not always obvious, and an extended interview with plant operators is usually required to chronologically arrange operator actions in each combination.

Analyzing dependency upfront for each accident sequence provides valuable "big picture" insights to the PRA analysist since each sequence is evaluated without the noise of operator actions appearing in cut sets from supporting systems (e.g., action to restore equipment to operation given a random failure).

Since the dependency analysis is performed upfront as part of the overall HRA, there is no need to quantify the model twice: first to identify combinations and second to perform the actual model quantification.

5.2.1.2 Approach Drawbacks

Many combinations from cut sets will be "missed" because operator actions that are not specific for a given sequence will not appear in the pre-quantification dependency analysis. For example, action to restore a component to operation following a random failure will not appear in the sequence-based dependency analysis because these actions are not sequence-specific. The "missed HFE" will appear in a cut set as:

IE * BE1-BEn * HFE1 * DepHFE2 * DepHFE3 * HFE4

Where HFE1, HFE2, and HFE3 are the sequence-specific operator actions that were included in the pre-quantification dependency analysis. HFE4 is the operator action that is not sequence-specific, thus, it was not identified in the dependency analysis and "showed up" in the cut set after quantification.

Concern

The concern with the not fully evaluated combinations is underestimation of the total probability of the resulting cut set if HFE4 in the example above is dependent on HFE3 (i.e., DepHFE4 should be used in this case).

Solution

To overcome this drawback, each sequence should be quantified separately with all HEPs set to a high value (e.g., 0.9). This will allow finding all operator actions that could potentially appear in the cut sets after the model quantification. After all the operator actions are identified, a review should be performed to identify actions that have a potential to alter the dependency analysis for the sequence-specific combinations (i.e., dependency level moderate, high, or complete). The assumption, which should be confirmed by testing and/or calculations, is that a low dependency (LD) for the non-sequence-specific HFEs will not appreciably alter the joint HEP of a cut set. HFEs with zero dependency (ZD) have no effect on the joint HEP.

The approach described above does not mean a full-dependency analysis to assign exact dependency level to each non-sequence HFE. Instead, a screening approach is proposed here. The higher dependency levels are warranted by certain conditions, described by PSFs. For the dependency analysis, PSFs include clues to initiate the action, available time to complete both HFEs, crew availability, proximity (location) of the actions, individual HFE stress level, etc. These PSFs are known from the independent HFE assessments. A systematic search through those PSFs will identify non-sequence-specific HFEs that have a potential to be dependent on the sequence-specific HFEs (total time window to perform all HFEs is too short, not enough crew members, etc.). Identified HFEs that may be highly dependent then moved to a full dependency analysis where they are analyzed (i.e., become DepHFE4 instead of HFE4 in the previous example).

5.2.2 Research Application – Model HRA Dependencies Similar to CCF

The SGTR ET in the generic PWR model was modified to logically build in all dependencies for each operator action. This approach was developed to eliminate the need for applying these dependencies after the cut sets are generated via post-processing rules. This approach was also developed to eliminate the potential of not properly accounting for all HRA dependencies due to some combinations that could be truncated.

This approach first solved the SGTR ET with all operator actions having a starting probability of 1.0 to identify all operator combinations. These identified operator action combinations were then assumed to dependent (just for research purposes) and a single dependent operator action basic event was developed for each combination. This single operator action was then properly modeled back into the logic structure. This single dependent operator action is similar to a common cause failure event that fails multiple components.

This approach identified three different issues that need to be addressed before it is a viable alternative:

- Remove cut sets that contain multiple dependent operator basic events that need to be removed because the combinations are not correct,
- Obtain cut sets of dependent operator actions along with component failures when only the independent operator action should appear,
- Provide additional scrutiny of the final cut sets and additional post-processing rules versus the current modeling process.

5.2.3 Artificial Intelligence and Machine Learning to Support Dependencies

Over decades of plant operation, NPPs have generated large amounts of data on the performance of plant structures, systems, and components (SSCs). Additionally, extensive models of plant performance and its response to normal operation and postulated transient and accident conditions have been developed. These models routinely are used to support plant licensing, operation, and maintenance activities. In the area of nuclear safety assessment and management, plant PRA models have been developed and are routinely applied to support various plant functions. Additionally, as the models have matured, they have expanded to address more diverse situations related to NPP risk and safety (e.g.,

inclusion of external plant hazards such as seismic, flooding, and high wind events, and the modeling of operator actions performed to respond to and mitigate plant events), as well as to support implementation of risk-informed plant programs (e.g., performance of online maintenance, Risk-Managed Technical Specification [RMTS] completion times, modification of surveillance frequencies, and specification of alternative treatments of safety grade plant SSCs).

Due to the volume of available data and the detailed modeling that has been performed, using this information can be extremely labor intensive and expensive. Additionally, there is a need to more effectively evaluate the models and data to obtain information needed to support decision making in a more-timely and cost-effective manner. Over the past several decades, advances in computational capabilities and analytical techniques now permit application of data analytic and machine learning (DA/ML) techniques to facilitate such analyses. However, these techniques are only now being explored for use within the nuclear industry with the initial applications just now being implemented. One possible reason for the slow adoption of DA/ML approaches appears to be nuclear industry's highly regulated nature. However, this situation is beginning to change with NPP owner/operators and regulatory authorities investigating and deploying applications that apply DA/ML techniques for a variety of applications.

A possible area where use of DA/ML approaches could be of great benefit is in querying plant PRA models to more efficiently. As indicated previously, plant PRA modes have evolved into large structures. When the need arises to obtain specific information (relevant data, accident sequences, human actions, results, etc.) the models currently need to be reviewed manually using highly specialized technical experts (i.e., PRA engineers) to obtain the relevant information, analyze and synthesize it, and then develop conclusions and actionable recommendations for presentation to responsible decision makers. This process is labor intensive and time consuming. Given the time constraints that often accompany plant decision making, conduct of these evaluations to meet the need is often challenging and expensive. Use of DA/ML approaches have the potential to greatly reduce the time and cost associated with obtaining data and information from PRA models to better support risk-informed decision making.

PRA models consist of ETs and FTs. ETs model sequences of events that may occur within an NPP the outcomes of which are classified as either an unacceptable outcome (e.g., core damage (CD) or large early release) or acceptable (e.g., no CD/no release of fission products). ET models are structured in a branching format similar to a decision tree. FTs model the likelihood of plant failure of plant SSCs that contribute to the various sequences that occur in one or more branches of the ET. FTs use a gated structure applying Boolean logic (e.g., AND and OR gates). From the perspective of computational architectures, ET and FT models constitute nodes (representative of the probability of occurrence of a particular event), which are connected to other events via edges. This represents the structure of an artificial neural network (ANN).

The structure of an ANN consists of one or more input nodes, one or more intermediate (hidden) layers, and one or more output nodes. Note that the simplest version of an ANN consists of a single input node and a single output node; this ANN is called a perceptron. In a typical ANN, various layers contain one or more nodes (which for the sake of discussion we call n) that transmit their respective n features (designated by the vector $\boldsymbol{X} = [x_1, x_2, \ldots, x_n]$) with edges of various weights (designated by the vector $\boldsymbol{b}$). An example of the structure and calculation processing of an ANN is shown in Figure 13.

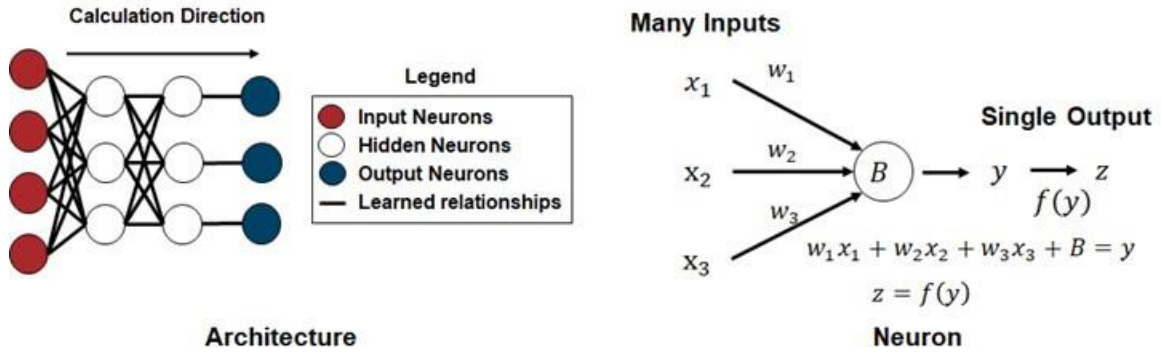

Figure 13. Simplified structure of an artificial neural network.

Based on the compatibility of the computational structures used in a plant PRA and an ANN, it is likely that use of processing techniques used in ANNs can enhance the functionality and usability of plant PRA models. One such application is in using ANN processing techniques to obtain relevant information from a plant PRA to support various plant decision making. As indicated previously, the current process to query the plant PRA model to obtain the desired information is manual, time consuming, and expensive. In addition, because the models are complex, it requires substantial effort and a detailed knowledge of the model to ensure all relevant portions of the model are investigated to ensure that all relevant information is identified for inclusion in the analysis of the specific issue being addressed. As an example of this, it is noted that FTs that model system failures can be applied multiple times (e.g., at different locations in the ET) for a particular accident sequence. A similar situation arises for the performance and evaluation of operator actions. Use of an ANN has the potential to automate a substantial amount of the effort employed in finding all the information that would be relevant to supporting plant risk-informed decision making. By automating this task, significant human labor could be saved resulting in freeing-up highly skilled resources to spend more time on analyzing the data and developing conclusions and recommendations. In addition, such an increase in efficiency would permit the conclusions and recommendations to the decision makers at an earlier point in time. This would provide plant decision makers with more time to evaluate the relevant information leading to better and more confident decisions.

5.2.4 Assign Joint HEP Minimum Value Automatically

It is the general practice for the industry performing HRA dependency analyses to assign a minimum joint HEP (a.k.a. floor joint HEP) to the combinations where the calculated joint HEP is too small. The joint HEP value is typically 1E-6. There may be limited cases where the joint HEP is not assigned based on specific justification for each case. The purpose of the minimum joint HEP is to account for uncertainties attributed to various unknown conditions and computational approximations. The SPAR dependency analysis process does not do this step.

The minimum joint HEP values are assigned to the majority of identified combinations with three or more HFEs since mathematically the product of three HEPs is usually below the minimum joint HEP, even after dependency levels are applied. This means that after dependency analysis is done and (e.g., 3,000 combinations are identified and processed), 80–90% of the combinations will be assigned a minimum joint HEP value instead of calculated one. This equates to 2,400–2,700 combinations that were already processed but will end up with the minimum joint HEP value anyway. In addition to the time spent, the recovery rule file becomes very long since it has an entry for each combination, to change the joint HEP to either the calculated one or to the minimum joint HEP. The issue becomes more significant with increased number of combinations (some models have over 10K combinations).

The proposed approach is to develop a recovery (post-processing) rule file that applies minimum joint HEPs to all the combinations that were not analyzed in the sequence-specific dependency analysis.

5.2.4.1 Approach Benefits

The largest benefit is that the HRA dependency analysis is accurate but generic enough due to the fact that the majority of combinations that are not risk-significant are assigned a single basic event with the probability of 1E-6 (or another justified joint HEP value). The generalization of the HRA dependency analysis provides an option of not performing it for each application unless changes are made to operator actions.

5.2.4.2 Approach Drawbacks

Concern

Some joint HEPs (JHEP) could potentially be underestimated (i.e., a calculated JHEP is higher than 1E-06), but 1E-06 is assigned automatically.

Solution

The solution proposed to overcome the joint HEP underestimation in Section 1 should be adequate in identifying HFEs with potential to appreciatively alter joint HEPs. Therefore, it is expected that very few, if any, combinations will have an underestimated joint HEP using this approach. However, trial evaluations should be performed to confirm this proposition.

5.2.4.3 Summary of Experimentation Results

As a proof of concept of the joint minimum HEP approach, the PRA model from a Westinghouse four loop PWR was used. For the demonstration performed in this work, the risk-significant combination of HEPs was identified based on the importance measures report for both CDF and LERF base case cut sets for the reference plant. A margin of 0.001 was assigned as a conservative measure to identify which combinations were risk-significant based on their Fussel-Vessely importance measure values. In this study, the risk-significant combinations retained their dependency probability values developed from the original plant HRA analysis, whereas all the remaining combinations were set to a floor value probability (i.e., a conservative value was assigned to combinations that have low impact) of 1.00E-06 and represented by a single basic event, "COMBINATION_floorJHEP," in the recovery rule file.

The seismic JHEP combinations were devised such that the combinations were used in later recovery commands of the recovery rule file. Thus, the following two sensitivities were performed:

- Floor JHEPs, excluding seismic JHEPs. Full power internal events (FPIE), flooding, and fire risk-significant combinations retained their original designation and value while the remaining combinations were assigned a floor value probability and represented by "COMBINATION_floorJHEP." The seismic combinations retained their original designation and values.
- Floor JHEPs, including seismic JHEPs. All FPIE, flooding, fire risk-significant combinations retained their original designation and value while the remaining combinations were assigned a floor value probability and represented by "COMBINATION_floorJHEP." The seismic combinations retained their original designation and risk-significant combinations retained their original values; however, non-risk-significant seismic combinations were assigned a floor value.

The results obtained are summarized in Table 7.

Assigning a floor value of 1E-06 to the non-risk-significant combinations reduced the overall CDF and LERF, as expected. The sensitivity changes, when compared to the base case results, are relatively small and the quantification times decreased for most of the sensitivity cases (note that quantification times vary with machine processing capacity). The second sensitivity, as expected, affected mostly the seismic results, as the non-risk-significant seismic JHEPs were set to a floor value of 1E-06.

A comparison between the sensitivity cases and the base case results are summarized in Table 8

As previously mentioned, the difference between the sensitivity cases and the base case is small, with the largest differences being in the order of 1E-06 for ΔCDF and 6.2E-07 for ΔLERF.

These results provide an initial validation of the approach that combining non-risk-significant JHEP combinations and/or setting them to a floor value is viable. Therefore, they support the conclusion that the proposed approach may be appropriate to reduce effort and time in evaluating HRA results in plant PRA models. Additionally, the decrease in quantification times could be beneficial for risk applications that require re-running the models in their evaluations. In particular, these savings could be substantial for use in application models where time limitations can impose significant constraints (e.g., plant CRM programs, risk-managed technical specifications) on plant staff.

Table 7. Assigning joint HEP minimum value automatically.

Sequence	CDF Results				LERF Results			
	Result	Truncation	Cut Sets	Quant Time	Result	Truncation	Cut Sets	Quant Time
Base Case								
All FPIE and External Hazards = TRUE	3.18E-05	1.00E-11	139249	10.7	1.16E-06	1.00E-12	159722	5.3
FPIE and Internal Flooding = TRUE	2.23E-06	1.00E-12	48080	1.6	9.94E-09	1.00E-15	203748	36
Fire = TRUE	2.47E-05	5.00E-12	96665	4.3	4.03E-07	5.00E-13	31851	10.9
Seismic = TRUE	6.16E-06	1.00E-11	101338	4.7	7.26E-07	1.00E-11	75846	4.4
Sensitivity Case - floor JHEPs (excluding seismic JHEPs)								
All FPIE and External Hazards = TRUE	3.08E-05	1.00E-11	128860	3.6	1.02E-06	1.00E-12	154967	5.1
FPIE and Internal Flooding = TRUE	2.08E-06	1.00E-12	38611	1.7	6.31E-09	1.00E-15	138203	29.7
Fire = TRUE	2.37E-05	5.00E-12	82356	3.6	2.50E-07	5.00E-13	24461	4.2
Seismic = TRUE	6.16E-06	1.00E-11	101338	0.30	7.26E-07	1.00E-11	75846	0.37
Sensitivity Case - floor JHEPs (including seismic JHEPs)								
All FPIE and External Hazards = TRUE	3.08E-05	1.00E-11	128424	5	1.01E-06	1.00E-12	154757	8.2
FPIE and Internal Flooding = TRUE	2.08E-06	1.00E-12	38611	2.2	6.31E-07	1.00E-15	138203	31.2
Fire = TRUE	2.37E-05	5.00E-12	82356	5.3	2.50E-07	5.00E-13	24461	3.5
Seismic = TRUE	6.13E-06	1.00E-11	100816	0.30	7.22E-07	1.00E-11	75693	0.36

Table 8. Dependency modeling analysis sensitivity case comparison.

	Sensitivity Case 1				Sensitivity Case 2			
	ΔCDF	Cut Sets Difference	ΔLERF	Cut Sets Difference	ΔCDF	Cut Sets Difference	ΔLERF	Cut Sets Difference
All FPIE and External Hazards = TRUE	1.00E-06	10389	1.40E-07	4755	1.00E-06	10825	1.50E-07	4965
FPIE and Internal Flooding = TRUE	1.50E-07	9469	3.63E-09	65545	1.50E-07	9469	6.21E-07	65545
Fire = TRUE	1.00E-06	14309	1.53E-07	7390	1.00E-06	14309	1.53E-07	7390
Seismic = TRUE	0.00E+00	0	0.00E+00	0	3.00E-08	522	4.00E-09	153

5.2.5 Capturing HRA Time Dependencies in Dynamic PRA

An option to solve some time-dependency issues and enable more accurate operator procedures is the use of dynamic PRA. However, one area that needs more research is how to apply PSFs to the models. An exercise was performed in this area using the EMRALD dynamic PRA software.

5.2.5.1 EMRALD Model

EMRALD is a dynamic PRA tool based on three phase discrete event simulation [emrald.inl.gov]. The sequences of a steam generator tube rupture (SGTR) ET were modeled using EMRALD diagrams with values to match the generic PWR SGTR model (Section 2.1.1). Times for actions were modeled using a normal distribution with the mean time determined from subject matter experts. A calculation using the mean and the PRA model failure rate was used to determine the standard deviation. If the sampled times for the tasks were more than the allowed time, then the task was considered failed as shown in Figure 14. Using these values, the EMRALD model matched the results of SAPHIRE within acceptable sampling range.

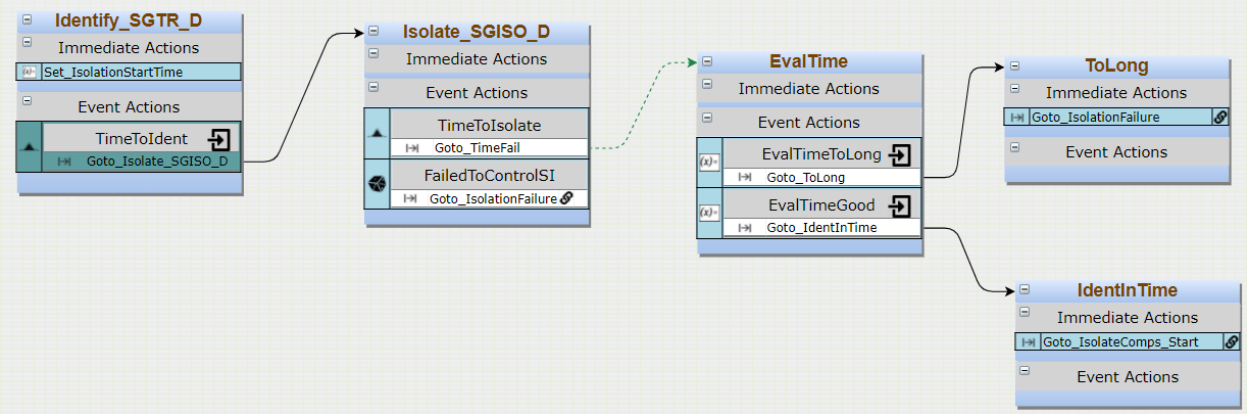

Figure 14. EMRALD diagram for the operator actions of the steam generator isolation steps.

While the model captures the operator actions and can credit for completing a task early to a subsequent task, it does not capture any dependencies or other PSFs. Some dependencies can be added to the EMRALD model using variables, such as tasks are performed close in time, time on duty, or stress. A cumulative variable in EMRALD can track the total amount of time spent in specific states, as shown by DBL_PrevStressTime in Figure 15. An event can then be added to specific states where the dependency may cause a failure as shown by OpStressFail_CC in Figure 15. The event uses the variable to calculate and sample if the failure occurred for that instance of the run.

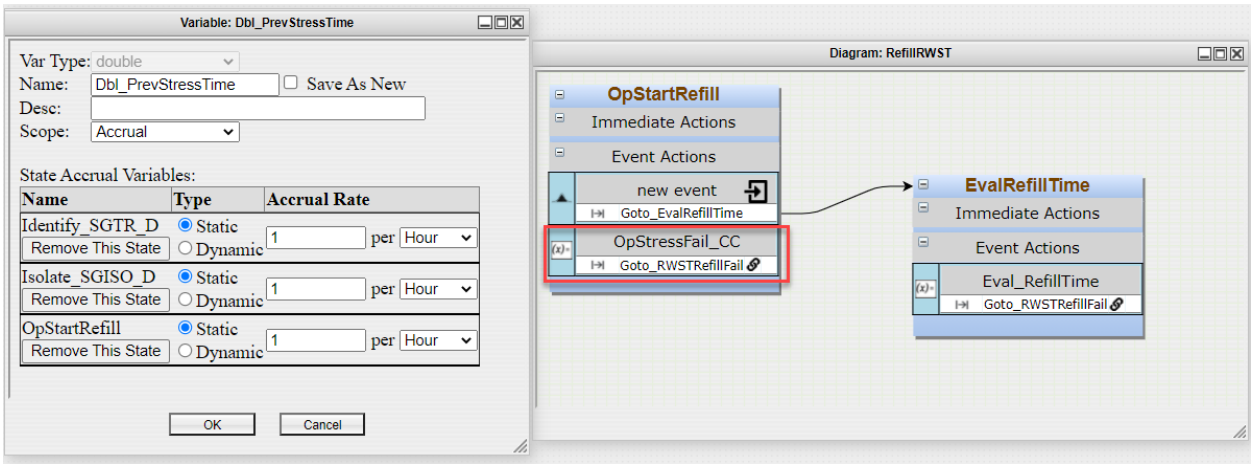

Figure 15. Variable used for stress dependency failure event.

While it is possible to model many HRA PSFs with existing EMRALD capabilities, it would likely require some complex scripts to be developed by the user. A better method would be to attach a dynamic HRA calculator that takes procedure steps and current simulation parameters and returns adjusted distribution parameters to sample on, and if applicable, a direct failure probability. Research is currently being done on how to implement a dynamic HRA calculator based on HUNTER [Boring et al. 2016] to couple with dynamic PRA tools.

6. CONCLUSION

Research was performed in the three areas identified as having near-term strategic benefit to PRA community:

1. Quantification Speed of Models.
2. Multi-Hazard Modeling Development, Maintenance, and Treatment.
3. Human Action Dependency Analysis.

A repository of benchmark reliability and risk models was created and will reside at Idaho State University. The repository was populated with a generic PWR SAPHIRE model and initial single-top versions of the ETs. We anticipate that this repository will be useful to provide baseline analysis timing results so that, as new approaches to quantification are proposed, the potential timing enhancements can be quantified.

Research and summarization of the current practices in each of the three areas above were performed and suggested improvements were proposed and tested. Many of the improvements have cross-over implications to the other areas. The quantification speed can be improved using minimum HEP factors, for instance, but can also be adversely affected by some of the improvements for multi-hazard modeling. These interactions need to be considered when proposing changes to realize the benefits in quantification speed expected in the use of HPC and container solving.

Quantification speed proposed improvements rely on three approaches: the use of HPC, container solving, and improved model management. The architecture required for using HPC with SAPHIRE is currently being developed. CAFTA can use HPC and enhancements to this process can be considered. Container solving is most promising with the ability to solve, store, and re-use results quickly. Container solving research is expected to move forward in FY 2022.

Multi-hazard modeling improvements beyond the use of container solving were identified as visualization improvements of results and methods to efficiently solve ET branches with a probability close to 1.0. Examples were presented for the visualization ideas. A method proposed to efficiently solve branches close to 1.0 was presented and tested on the benchmark model with successful results.

Human action dependency improvements rely on proper manipulation of HEP combinations. A proposal was presented to find these in an automated fashion through AI/ML. An exercise using a benchmark CAFTA model was successful in realistically capturing dependency while using a floor value for identified events.

7. REFERENCES

Akers, S. 1978. "Binary Decision Diagrams." *IEEE Transactions on Computers*. Vol. C-27. No. 6. June.

Andrews, J. D. and S. J. Dunnett. 2000. "Event-Tree Analysis Using Binary Decision Diagrams." *IEEE Transaction on Reliability*. Vol. 49. No. 2. June.

Boring, R. L. 2015. "A Dynamic Approach to Modeling Dependence between Human Failure Events." *Proceedings of the 2015 European Safety and Reliability (ESREL) Conference*: 2845–2851.

Boring, R., D. Mandelli, M. Rasmussen, S. Herberger, T. Ulrich, K. Groth and C. Smith. 2016. "Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER): A Framework for Computational-based Human Reliability Analysis." *13th International Conference on Probabilistic Safety Assessment and Management (PSAM 13)*. Paper A-531: 1–7.

Boring, R., J. Park and T. Mortenson. 2021. "Formative vs. Summative Dependence in Human Reliability Analysis." *Lecture Notes in Network and Systems*. Vol. 262: 62–69.

Corvino, J. 2020. *Rapid Quant*. presentation at EPRI's Annual IRT User's Group Meeting. January 15.

Friedenthal, S., A. Moore, and R. Steiner, 2008, A Practical Guide to SysML: The Systems Modeling Language, Morgan Kaufmann.

GIT, git-scm.com, GIT open source fast version control system, accessed September 2021.

Hecht, M., E. Dimpfl, J. Pinchak, 2014 , "Automated Generation of Failure Modes and Effects Analysis from SysML Models," IEEE International Symposium on Software Reliability Engineering Workshops.

INL. EMRALD. Emrald.inl.gov., official software website for the EMRALD dynamic PRA software program, accessed September 2021.

INL. "Raven." raven.inl.gov., official software website for the RAVEN uncertainty quantification, regression analysis, and dynamic PRA, data analysis, and model optimization framework software program, accessed September 2021.

Mandelli, D., C. Wang, C. Parisi, D. Maljovec, A. Alfonsi, Z. Ma, C. Smith, 2020, "Linking Classical PRA Models to a Dynamic PRA," Annals of Nuclear Energy, 149.

Mandelli, D., A. Alfonsi, T. Aldemir, 2021, "Generation of Event Trees and Fault Trees: A Model-Based Approach," Proceedings of PSA 2021 Conference.

Miller, A., S. Hess, and C. Smith. 2020. R&D Roadmap to Enhance Industry Legacy Probabilistic Risk Assessment Methods and Tools. INL/EXT-20-59202.

Mortenson, T. and R. Boring. 2021. "Is Dependency in Human Reliability Analysis a Real Phenomenon? Refining the Dependency Concept Through Research." *Lecture Notes in Network and Systems*, Vol. 262: 55–61.

Park, J. and R. L. Boring. 2021. “Identification of Performance Shaping Factors Affecting Subsequent Human Actions for Dependence Assessment in Human Reliability Analysis.” *Lecture Notes in Network and Systems.* Vol. 262: 47–54.

Rauzy, A. and Y. Dutuit. 1997. “Exact and truncated computations of prime implicants of coherent and non-coherent fault trees within Aralia.” *Reliability Engineering and System Safety*. Vol. 58: 127–144.

Sinnamon, R. M. and J. D. Andrews. 1997. “Improved Accuracy in Quantitative Fault Tree Analysis.” *Quality and Reliability Engineering International.* Vol. 13: 285–292.

Swain, A. D. and H. E. Guttmann. 1983. *Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications*. NUREG/CR-1278. U.S. Nuclear Regulatory Commission.

U.S. NRC. 1975. *The Reactor Safety Study*. https://www.nrc.gov/docs/ML1622/ML16225A002.pdf. WASH-1400.

U.S. NRC. 2011. *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 8: Technical Reference.* NUREG/CR-7039. Vol. 2. https://www.nrc.gov/reading-rm/doc-collections/nuregs/contract/cr7039/v2/index.html.